UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

UNE ANALYSE THEORIQUE ET NUMERIQUE DU PROBLEME DU CHOIX OPTIMAL D'UN RESEAU DE TRANSPORT SANS CONGESTION

par

René Dionne

Cette recherche appliquée a été subventionnée en partie par "Ford Motor Company of Canada".

Université de Montréal
Publication # 198 - Département de Montréal
Publication # 27 - Centre de recherche sur les transports
Octobre 1974

TABLE DES MATIERES

Chapitre / section			page
		SOMMAIRE	vi
I		INTRODUCTION	1
II		ETUDE ET EXTENSION D'UN ALGORITHME DE MURCHLAND	15
	2.1	Le cas orienté	16
	2.1.1	Diminution de la longueur d'un arc	17
	2.1.2	Augmentation de la longueur d'un arc	26
	2.2	Le cas symétrique	32
	2.2.1	Diminution de la longueur d'une arête	32
	2.2.2	Augmentation de la longueur d'une arête	37
III		RECHERCHE D'UN ALGORITHME EFFICACE POUR LE PROBLEME DU CHOIX OPTIMAL D'UN RESEAU	42
	3.1	Introduction	42
	3.2	Présentation de l'algorithme initial	43
	3.2.1	Notation	43
	3.2.2	Définition du problème de choix optimal d'un réseau	46
	3.2.3	L'arborescence associée	47
	3.2.4	La borne inférieure I^P	51
	3.2.5	Le critère de choix de la variable séparatrice	56
	3.2.6	La solution initiale	57
	3.2.7	Définition de l'algorithme de Hoang	58
	3.2.8	Quelques remarques d'ordre pratique pour l'implantation de l'algorithme	63

	3.3	Une approche expérimentale à la recherche d'un algorithme plus efficace	67
	3.3.1	Introduction	67
	3.3.2	Quelques remarques et notation supplémentaire	69
	3.3.3	Comparaison de l'option 1 et de l'option 2	72
	3.3.4	Comparaison de l'option 2 et de l'option 3	75
	3.3.5	Etude paramétrique de l'option 3	79
	3.3.6	Comparaison de l'option 3 et de l'option 4	83
	3.3.7	Influence de la solution initiale sur le temps de calcul	89
	3.4	Conclusion	92
IV		QUELQUES HEURISTIQUES POUR LE PROBLEME DU CHOIX OPTIMAL D'UN RESEAU	94
	4.1	L'heuristique MGPM	95
	4.2	L'heuristique MQJ	96
	4.3	L'heuristique MQJM	98
	4.4	L'heuristique MQO	99
	4.5	Les heuristiques MQJM et MQO vues comme alternatives aux méthodes optimales	101
V		EXTENSIONS	106
	5.1	Généralisations	106
	5.2	Modifications à apporter à l'heuristique MQJM	110
	5.3	Modifications à apporter à l'option 3	111
	5.4	Modifications à apporter à l'heuristique MQO	115
VI		CONCLUSION	116

APPENDICE I 119

APPENDICE II 149

REMERCIEMENTS 157

BIBLIOGRAPHIE 158

SOMMAIRE

Le problème du choix optimal d'un réseau consiste à déterminer, parmi un ensemble de projets de construction de routes, lesquelles routes on va effectivement construire pour obtenir un réseau qui assure le meilleur service possible tout en respectant un budget limitatif. Le phénomène de la congestion n'est pas considéré ici. La présente thèse étudie ce problème du point de vue théorique; elle propose aussi des algorithmes et elle en fait l'évaluation numérique.

Le meilleur algorithme optimal que nous ayons mis au point pour résoudre le problème du choix optimal d'un réseau se révèle plus efficace que les algorithmes publiés antérieurement. Deux nouvelles heuristiques sont aussi proposées pour résoudre le même problème; elles demandent peu de temps de calcul et produisent une solution réalisable étonnamment proche de la solution optimale. Cet algorithme optimal et ces deux heuristiques, de même que l'étude théorique sur laquelle ils sont fondés, constituent nos principales contributions à la discipline de la recherche opérationnelle.

La présente thèse fait aussi l'étude d'un algorithme initialement proposé par Murchland (1970) pour calculer les nouvelles longueurs des plus courts chemins après modification de la longueur d'un seul arc dans un réseau. Les résultats de cette étude trouvent une application immédiate dans la résolution du problème du choix optimal d'un réseau.

CHAPITRE I

INTRODUCTION

Dans le développement d'un réseau de transport, le planificateur doit prendre des décisions nombreuses et interdépendantes quant aux améliorations à apporter au réseau. Ces améliorations visent à répondre aux demandes futures de déplacements entre divers points du réseau; elles sont de deux types:

- on peut augmenter l'efficacité de certains tronçons qui existent déjà dans le réseau (en élargissant une route, en appliquant une politique de voies à sens unique, etc.)
- et on peut ajouter de nouvelles voies.

C'est surtout à ce deuxième type d'améliorations que nous nous intéresserons dans la présente thèse.

Le problème que nous étudierons est celui du choix optimal d'un réseau et il s'énonce comme suit: étant donné un ensemble de projets de construction de routes, déterminer quelles routes il faut effectivement construire pour obtenir le réseau optimal suivant certains critères, si les sommes à dépenser sont limitées par un budget. Ces critères ou mesures d'efficacité peuvent prendre diverses formes, mais ils sont habituellement assimilés au coût d'utilisation du réseau, qui n'est autre que la somme de tous les coûts individuels encourus par les utilisateurs du réseau. Les projets de construction auront été préalablement déterminés en tenant compte de tous les facteurs pertinents: utilisation du sol, facteurs so-

ciaux, facteurs écologiques, etc. S'il faut améliorer le réseau, c'est que, dans un avenir plus ou moins rapproché, le réseau ne pourra plus suffire à la demande, soit parce qu'il deviendra congestionné, soit parce qu'il ne desservira pas assez bien certaines régions en plein développement. On représente le réseau à l'aide d'un graphe et on considère que les déplacements se font entre certaines paires de sommets de ce graphe, appelées paires origine-destination (paires O-D). Si on ne tient pas compte de la congestion, le temps de parcours d'un arc est une constante. Mais si l'on veut représenter la congestion sur un arc, on a habituellement recours à une fonction de temps de parcours. Une telle fonction est convexe croissante; son évaluation donne le temps de parcours en fonction de la charge (ou flot) sur l'arc. Pour chaque paire O-D, la demande de déplacements aura été préalablement estimée pour l'année-cible (horizon de planification). Le réseau optimal doit satisfaire sans congestion aux demandes prévues.

Dans la pratique actuelle, la façon courante de résoudre le problème de choix optimal d'un réseau consiste à ne conserver que quelques projets de réseau jugés intéressants ou prometteurs et à les évaluer séparément par une méthode d'affectation du trafic pour ne conserver que le meilleur réseau. Cette approche néglige d'évaluer un grand nombre de réseaux et elle peut facilement laisser échapper le réseau optimal.

Un algorithme efficace qui fournirait la solution optimale au problème du choix optimal d'un réseau trouverait application immédiate dans le contexte du développement des réseaux de transport. Au cours de la dernière décade, plusieurs chercheurs ont travaillé en ce sens. Plusieurs

modèles ont été proposés, que l'on peut diviser en deux classes, suivant le principe utilisé pour répartir le trafic sur les différents arcs du réseau:

- les modèles descriptifs*, ainsi nommés parce qu'ils prétendent décrire la réalité en postulant que l'utilisateur d'un réseau choisit son itinéraire de façon à minimiser son coût (ou temps) de parcours. Cette situation correspond au premier principe de Wardrop (1952), qui s'énonce ainsi: toutes les routes utilisées entre une origine et une destination ont le même temps de parcours et toutes les routes non utilisées ont un temps de parcours plus grand.
- les modèles normatifs*, dans lesquels les itinéraires des utilisateurs d'un réseau sont déterminés de façon à optimiser globalement l'utilisation d'un réseau pour l'ensemble de ses utilisateurs. Tout se passe alors comme si une autorité centrale pouvait régir le trafic. Cette façon de procéder correspond au deuxième principe de Wardrop. Les deux principes de répartition du trafic ne donnent des flots équivalents sur les arcs que dans le cas des réseaux opérant en deça du niveau de congestion (Jorgensen (1963)). L'écart entre les deux types de flots peut cependant être réduit par des mesures incitatives comme, par exemple, la tarification (Dafermos (1973)).

Parlons d'abord des modèles normatifs. En général, ce sont des modèles de programmation linéaire. Le réseau maximal (celui qui inclut

*Note: Nous parlons simplement de "modèles descriptifs (ou normatifs)" pour éviter d'avoir à utiliser l'expression plus juste mais encombrante de "modèles pour le problème du choix optimal d'un réseau basés sur le modèle descriptif (ou normatif) d'affectation du trafic".

tous les projets de construction) et les demandes entre les paires O-D y sont décrits par l'intermédiaire de contraintes linéaires de conservation de flot (équations de flot), d'une façon similaire à celle employée pour le problème de flot de plusieurs produits ("multicommodity network flow", cf. la formulation sommet-arc de Tomlin (1966)). Les équations de flot constituent d'ailleurs l'essentiel de ces modèles.

Carter et Stowers (1963) ont proposé un tel modèle. Suivant une pratique répandue, ils décrivent le temps de parcours sur chaque arc par une fonction convexe croissante du flot sur cet arc (cas avec congestion); et pour préserver la linéarité de leur modèle, ils approximent cette fonction par une fonction linéaire par morceaux. Ils s'intéressent à l'amélioration d'un réseau par l'augmentation de la capacité des arcs; à cette fin, ils emploient pour chaque arc une variable indiquant l'augmentation éventuelle de la capacité de l'arc. L'objectif visé est la minimisation de la somme des coûts de construction et des temps de parcours, en respectant une contrainte budgétaire. Ils obtiennent ainsi un programme linéaire entièrement en variables continues.

Roberts et Funk (1964) font à juste titre remarquer que l'addition de capacité à un arc ne se fait pas de façon continue, mais discrète. Ils proposent un modèle de programmation linéaire partiellement en nombres entiers pour le problème d'addition d'arcs nouveaux dans un réseau. Leur formulation repose, comme chez Carter et Stowers, sur les équations de flot; et, à chaque projet de construction d'un arc, correspond une variable bivalente indiquant si oui ou non le projet sera construit. Une contrainte linéaire limite le coût de construction au budget alloué. Le

coût d'utilisation d'un arc est une constante assimilable au temps de parcours de l'arc. Le flot sur un arc est limité par une capacité seulement pour les arcs existant avant construction. Et la fonction économique à minimiser est linéaire (coûts de construction et d'utilisation). Pour bien mettre en évidence la nature discrète du problème, Roberts et Funk commencent par résoudre un petit problème (7 sommets, 4 projets) par la méthode du simplexe; ils jugent les résultats insatisfaisants et utilisent alors un algorithme de programmation linéaire en nombres entiers.

Hershdorfer (1965) aussi traite le problème du choix optimal d'un réseau par la programmation linéaire en nombres entiers. Dans sa formulation, les variables de décision sont le nombre de voies à ajouter aux arcs existants. Des contraintes (linéaires) supplémentaires permettent de décider de l'orientation optimale des flots (politique de voies à sens unique). Quant à la fonction économique, Hershdorfer fait ressortir que, d'un point de vue conceptuel, il est fort discutable de poser une fonction économique qui soit la somme des coûts d'utilisation et des coûts de construction, étant donné que ces deux types de coûts sont essentiellement différents quant aux unités dans lesquels on les exprime (unités de temps pour les uns, unités monétaires pour les autres). Il discute trois variantes pour la fonction économique:

- minimiser la somme des coûts de construction et d'utilisation
- minimiser les coûts d'utilisation en respectant un budget donné
- minimiser l'investissement pour une réduction donnée des temps de parcours.

Scott (1967) aussi propose un programme linéaire avec des variables entières. Son objectif consiste à déterminer le réseau non congestionné à coût d'utilisation minimal tout en restant dans les limites d'un budget.

Tous les modèles normatifs mentionnés jusqu'ici présentent deux inconvénients majeurs:

- résoudre un modèle normatif fournit une solution avantageuse pour l'ensemble de la société des utilisateurs. Cependant, dans la vie réelle, il semble plutôt que chaque utilisateur travaille à minimiser son temps de parcours personnel ...
- la taille des programmes linéaires en jeu devient facilement énorme. Qu'il suffise de mentionner que les équations de flot à elles seules font intervenir $n\,p$ équations et $(m + 1)p$ variables, où l'on définit

  m = nombre d'arcs

  n = nombre de sommets

  p = nombre de paires O-D .

  (Pour l'établissement de ces nombres, cf. Steenbrink (1974), page 16). Pour la valeur de p , une borne supérieure est $n(n - 1)$. Pour fixer un peu les idées, considérons un réseau à 10 sommets et 20 arcs: dans ce cas, on pourrait obtenir jusqu'à 900 équations et 1890 variables. Et nous ne tenons pas compte des contraintes autres que les équations de flot. Bien sûr, l'utilisation de la méthode de décomposition de Dantzig et Wolfe (cf. par exemple Simonnard (1962)) est toute indiquée ici, mais il est significatif de noter que peu de résultats numériques apparaissent dans la

littérature pour les modèles normatifs: les plus grands réseaux traités numériquement comportent 12 sommets (Hershdorfer (1965)).

Dans les modèles descriptifs, la répartition du trafic sur les arcs ne peut plus se faire implicitement par l'intermédiaire des équations de flot. Il faut utiliser des relations qui décrivent le comportement des utilisateurs conformément au premier principe de Wardrop (principe des temps de parcours égaux pour tous les utilisateurs entre les deux sommets d'une paire O-D), ce qui exige le calcul de tous les plus courts chemins. Pour cette raison, et aussi à cause de la présence de variables entières, il paraît indiqué d'énumérer et d'évaluer une à une les diverses solutions et de chercher parmi elles la solution optimale. Une énumération complète s'avérerait prohibitive: en effet, si on a m projets et, pour chacun de ces projets, seulement deux possibilités (construire ou non), on obtient 2^m réseaux possibles. On a donc recours à une technique d'énumération implicite (procédure d'exploration par séparation et évaluation).

Une telle procédure consiste à partitionner l'ensemble de toutes les solutions (réalisables et non réalisables) en sous-ensembles disjoints. Pour chacun de ces sous-ensembles, on calcule (dans le cas d'un problème de minimisation) une borne inférieure, qui est par définition plus petite ou égale à la valeur de la meilleure solution du sous-ensemble. Chacun de ces sous-ensembles est examiné pour tenter de l'éliminer, ce qui se produit dans trois cas:

- il n'y a qu'une seule solution dans le sous-ensemble. Si cette solution est réalisable et si elle surclasse la meilleure solution

courante, elle remplace celle-ci.

- si le sous-ensemble ne contient aucune solution réalisable
- si la borne inférieure calculée pour le sous-ensemble est plus grande ou égale à la valeur de la meilleure solution courante.

Tout sous-ensemble non éliminé est lui-même séparé par une partition. Lorsque tous les sous-ensembles générés ont été éliminés, la meilleure solution courante est la solution optimale. On trouvera au chapitre III de la présente thèse (section 3.2.3) un exposé plus complet sur les procédures d'énumération implicite.

Plusieurs chercheurs ont formulé un modèle descriptif pour le problème du choix optimal d'un réseau et ont proposé de le résoudre par une procédure d'énumération implicite: Ridley (1968), Ochoa-Rosso (1968), Ochoa-Rosso et Silva (1968), Scott (1969), Chan (1969), Hoang (1971), Bruynooghe (1972), Leblanc (1973) et Boyce *et al.* (1973).

La technique d'énumération implicite laisse beaucoup de latitude quant à l'élaboration d'une procédure spécifique. La façon de calculer les bornes et la façon de déterminer l'ordre dans lequel les sous-ensembles sont examinés varient beaucoup d'un auteur à l'autre. Rendre compte des procédures utilisées par chacun des auteurs mentionnés au paragraphe précédent exigerait, pour rendre justice à chacun, que nous allions dans le détail et dépasserait de beaucoup le cadre de la présente introduction. Nous nous contenterons de préciser à quelle variante du problème du choix optimal d'un réseau chacun d'eux s'intéresse.

En général, ils cherchent à déterminer le réseau à coût d'utilisa-

tion minimal (sauf Chan, qui minimise l'investissement tout en maintenant le coût d'utilisation en deça d'un certain niveau, et Bruynooghe, qui ajoute le coût de construction au coût d'utilisation); ce réseau à coût minimal doit être obtenu en investissant une somme fixée d'avance (budget) dans des projets de construction (il s'agit de nouvelles routes, sauf pour Ridley qui considère des améliorations à apporter aux arcs déjà en place pour réduire le temps de parcours. Leblanc considère les deux possibilités). Chaque fois qu'il est nécessaire de connaître la valeur de la fonction économique pour un réseau énuméré, une affectation du trafic est effectuée sur le réseau en conformité avec le premier principe de Wardrop. Pour Ridley, Scott, Hoang et Boyce *et al.*, cette affectation se fait ainsi: pour chaque paire O-D, toute la demande passe par le plus court chemin entre les deux sommets de la paire O-D et le coût de parcours d'un arc pour chaque usager est une constante assimilable au temps de parcours de l'arc. Cela suppose qu'il ne peut y avoir de congestion sur aucun arc du réseau (contexte régional ou rural). Les autres (Ochoa-Rosso, Ochoa-Rosso et Silva, Chan, Bruynooghe et Leblanc) traitent le cas où il y a congestion et ils utilisent tous des fonctions convexes croissantes qui donnent le temps de parcours d'un arc en fonction du flot sur cet arc. Le réseau optimal doit satisfaire (sans congestion, s'il y a lieu) aux demandes de flot entre les paires O-D (sauf pour Hoang, qui considère les demandes égales entre toutes les paires O-D et, par suite, n'en tient pas compte). Certains auteurs (Scott, Hoang, Boyce *et al.*) considèrent qu'il n'y a aucune route de construite au départ et, par conséquent, ils doivent assurer la connexité du réseau optimal.

Tout comme les modèles normatifs, les modèles descriptifs fournissent un cadre théorique intéressant pour le problème du choix optimal d'un réseau, mais ils sont assez limités au point de vue numérique. Les algorithmes d'énumération implicite proposés pour résoudre les modèles descriptifs ont comme principal désavantage qu'il est nécessaire de procéder à une affectation de trafic chaque fois qu'on veut évaluer un réseau énuméré; et, même si une faible proportion de tous les réseaux possibles est énumérée, il reste un grand nombre de réseaux à évaluer. Quelques chercheurs (Ochoa-Rosso et Silva, Scott, Hoang, Leblanc et Boyce *et al.*) rapportent des résultats numériques obtenus avec leur algorithme respectif. Les résultats les plus intéressants nous paraissent être:

- pour le cas sans congestion, ceux de Hoang (temps de calcul variant de 13.5 à 188 secondes, suivant le budget, pour un réseau symétrique à 9 sommets et 20 arêtes-projets) et ceux de Boyce *et al.* (temps de calcul de l'ordre d'une seconde jusqu'à plusieurs centaines de secondes suivant que le budget est peu ou très restrictif, pour un réseau symétrique à 10 sommets et 45 arêtes-projets)
- pour le cas avec congestion, celui de Leblanc (un temps de 2 minutes 16 secondes pour 24 sommets, 76 arêtes, mais 5 projets seulement).

Puisque la recherche d'une solution optimale par une méthode exacte s'avère très coûteuse en temps de calcul, certains chercheurs se sont tournés vers des méthodes heuristiques. De telles méthodes constituent un compromis: en échange d'un temps de calcul plus court, elles sacrifient l'optimalité, mais prétendent tout de même donner une solution proche de

l'optimum (sinon égale à l'optimum).

Scott (1969) a proposé deux heuristiques d'une grande simplicité. Grosso modo, la première consiste à commencer avec l'arbre à coût (de construction) minimal (minimum spanning tree), à ajouter ensuite l'arc qui fait le plus diminuer la fonction économique, puis à itérer aussi longtemps qu'on ne dépasse pas le budget; la deuxième méthode part du réseau maximal (tous arcs inclus), supprime ensuite l'arc qui induit la plus faible augmentation de la fonction économique, et poursuit en itérant aussi longtemps que le coût de construction dépasse le budget. Hoang (1971) a proposé une heuristique équivalente à cette deuxième méthode (cf. méthode des C_j, section 3.2.6 de la présente thèse). Barbier (1966) suggère une méthode itérative qui consiste à effectuer d'abord une affectation sur le réseau maximal, puis à déterminer par inspection de chaque arc s'il est possible de réaffecter le flot sur cet arc à d'autres arcs de façon à diminuer la valeur de la fonction économique et à supprimer chaque arc pour lequel cela est possible. Après inspection de tous les arcs, on obtient un nouveau réseau pour lequel on répète le processus, si c'est possible. Mentionnons encore les heuristiques de Billheimer et Gray (1973) et Steenbrink (1974). Billheimer et Gray proposent un algorithme qui emploie alternativement des critères d'exclusion et de réinsertion des arcs dans le réseau; leur algorithme converge vers un optimum local et il a été testé avec un problème de 68 sommets et 476 arcs (temps de calcul d'environ 3 minutes). Quant à Steenbrink, il suggère une affectation par étapes: à chaque étape, on affecte une partie de chacune des demandes O-D à la route qui a le coût marginal le moins élevé.

Cette dernière méthode (Steenbrink (1974), chapitre 5) a été utilisée dans une étude d'optimisation du réseau routier des Pays-Bas (2000 sommets, 6000 arcs, 351 paires O-D). Enfin, certains auteurs (par exemple, Loubal (1967) et Stairs (1968)) ont suggéré de combiner les capacités intellectuelles de l'homme aux capacités de calcul de l'ordinateur pour résoudre le problème du choix optimal d'un réseau de façon interactive; Stairs rapporte avoir étudié ainsi deux réseaux, dont le plus grand comporte 35 sommets et 10 projets de construction.

Pour compléter ce bref bilan que nous avons voulu dresser de l'état actuel de la recherche dans le domaine du choix optimal d'un réseau, nous ne pouvons omettre de mentionner l'excellente monographie de Steenbrink (1974) sur la question de l'optimisation des réseaux de transport, qui englobe le problème du choix optimal d'un réseau. Avant Steenbrink, Schwartz (1968) et Stairs (1968) avaient passé en revue la littérature sur le problème du choix optimal d'un réseau, mais leur travail est incomplet, ne serait-ce que parce qu'il date de 1968.

La présente thèse vise trois objectifs:

- faire le point sur la situation de la recherche sur le problème du choix optimal d'un réseau (dans la présente introduction)
- à partir d'un modèle descriptif, mettre au point un algorithme qui soit le plus efficace possible, pour le problème de choix optimal d'un réseau dans le cas où il n'y a pas de congestion, afin de pouvoir traiter des problèmes de plus grande taille que ceux traités jusqu'à maintenant
- proposer de nouvelles heuristiques pour le même problème.

Le plan de la thèse est le suivant. Après l'introduction, le deuxième chapitre est consacré à l'étude et à l'extension d'un algorithme de Murchland (1970). Cet algorithme sert à calculer la matrice des longueurs des plus courts chemins après modification d'un seul arc dans un réseau. Nous étudions d'abord l'algorithme d'un point de vue théorique et nous en proposons une version légèrement améliorée. Nous adaptons ensuite l'algorithme pour le cas d'un réseau symétrique et c'est sous cette nouvelle forme que nous l'utilisons dans le chapitre suivant. Nous avons aussi étudié l'algorithme d'un point de vue numérique. D'ailleurs, dans la présente thèse, une proportion importante de notre travail possède une orientation résolument expérimentale; trop souvent hélas, les chercheurs se contentent d'une étude théorique, sans résultats expérimentaux convenables.

Le troisième chapitre décrit comment nous procédons pour mettre au point un algorithme efficace pour le problème du choix optimal d'un réseau sans congestion. Notre travail s'appuie sur de nombreux résultats numériques obtenus à l'aide d'un programme expérimental FORTRAN (reproduit dans l'appendice I). Nous estimons que les résultats obtenus indiquent que l'algorithme mis au point surclasse les algorithmes publiés jusqu'à maintenant.

Le chapitre IV est consacré à l'étude de quelques heuristiques. L'évaluation numérique de ces heuristiques nous permet d'en faire ressortir deux comme particulièrement intéressantes par le fait qu'elles fournissent une solution étonnamment proche de la solution optimale, bien qu'elles exigent peu de temps de calcul.

Dans un cinquième chapitre, nous donnons une formulation du problème du choix optimal d'un réseau plus générale que la formulation employée aux chapitres III et IV, et nous étudions les modifications que cette formulation entraîne sur les algorithmes proposés aux chapitres III et IV.

Le chapitre VI constitue une brève conclusion sur notre travail.

CHAPITRE II

ETUDE ET EXTENSION D'UN ALGORITHME DE MURCHLAND

En traitant le problème du choix optimal d'un réseau, on se trouve souvent dans la situation suivante: ayant modifié la longueur d'un seul arc d'un réseau, on aimerait connaître la nouvelle matrice des longueurs des plus courts chemins sans la recalculer au complet par l'algorithme de Floyd-Murchland (c'est-à-dire l'algorithme défini par Floyd (1962), puis redécouvert et exploité par Murchland; pour plus de détails, cf. Dreyfus (1969)).

Murchland (1970) a proposé un algorithme pour calculer cette nouvelle matrice des longueurs des plus courts chemins. Le présent chapitre rend compte d'une étude qui a été menée dans le but de connaître l'efficacité de ce dernier algorithme (Murchland (1970), p. 39 et p. 43). Il présente d'abord l'algorithme pour le cas orienté, selon un point de vue qui a permis d'apporter une amélioration à l'algorithme tel que défini par Murchland. Il traite ensuite le cas non orienté, ce qui n'a pas été fait par Murchland. Pour chacun de ces deux cas, sont présentés des résultats numériques obtenus à l'aide de l'ordinateur CDC CYBER 74 du Centre de Calcul de l'Université de Montréal.

2.1 Le cas orienté

Le problème à résoudre s'énonce comme suit. Pour un réseau donné, on connaît la longueur de tous les plus courts chemins entre chaque paire de points. On modifie la longueur d'un arc (ou encore on ajoute ou on supprime un arc). On doit calculer les nouvelles longueurs des plus courts chemins, mais on serait mal avisé de recalculer les longueurs de tous les plus courts chemins. On doit définir un algorithme qui permette de repérer les plus courts chemins qui sont effectivement modifiés et qui ne refait le calcul que pour ceux-ci. Il faut bien entendu garder le nombre d'opérations à un minimum.

Le réseau est défini par la donnée d'un graphe $G = (N,A)$ et d'une matrice D .

$N = \{1,2,\ldots,n\}$ = ensemble des points ou sommets du graphe G

A = ensemble des arcs du graphe G

$D = (d_{ij})_{nxn}$, où d_{ij} = longueur de l'arc du sommet i au sommet j

d'_{ij} = valeur modifiée de d_{ij} .

Lorsqu'il n'y a pas d'arc de i à j, on conviendra de poser $d_{ij} = M$ (M grand; par exemple, $M = n \cdot \max_{(i,j)\in A} d_{ij}$). Ainsi on pourra assimiler le cas où un arc est ajouté ($d'_{ij} < d_{ij} = M$) au cas où la longueur d'un arc est simplement réduite ($d'_{ij} < d_{ij} < M$). On traitera la suppression d'un arc ($d_{ij} < d'_{ij} = M$) de la même façon que son simple allongement ($d_{ij} < d'_{ij} < M$).

On connaît (ou on peut obtenir par l'algorithme de Floyd-Murchland) la matrice des plus courts chemins $C = (c_{ij})_{nxn}$, où c_{ij} désigne la longueur du plus court chemin de i à j. A noter que $c_{ij} \geq M$ signifie qu'il n'y a pas de chemin de i à j.

On supposera aussi que $d_{ii} = c_{ii} = 0$, $i \in N$

et $d_{ij} > 0$, $i \neq j$.

Enfin, rappelons que le réseau étudié ici est orienté (d_{ij} peut différer de d_{ji}).

2.1.1 <u>Diminution de la longueur d'un arc</u>

L'arc modifié est (j,k) et sa nouvelle longueur, d'_{jk}.

$$0 < d'_{jk} < d_{jk} \leq M .$$

Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changement dans C. On supposera donc ici que $d'_{jk} < c_{jk} \leq d_{jk}$.

On doit déterminer toutes les paires (i,ℓ) de sommets pour lesquelles $c_{i\ell}$ doit être modifié (nous parlerons plus simplement des paires (i,ℓ) à modifier). Avant la modification de la longueur de l'arc (j,k), deux cas sont possibles pour chaque paire (i,ℓ) : ou bien le plus court chemin de i à ℓ passe par l'arc (j,k) (1er cas) ou bien il n'y passe pas (2e cas).

1er cas

——→ représente un arc

- - - - - → représente un plus court chemin

Le plus court chemin de i à ℓ passe par l'arc (j,k) et se décompose en trois parties:

1) le plus court chemin de i à j, de longueur c_{ij}

2) le plus court chemin de j à k, de longueur d_{jk}

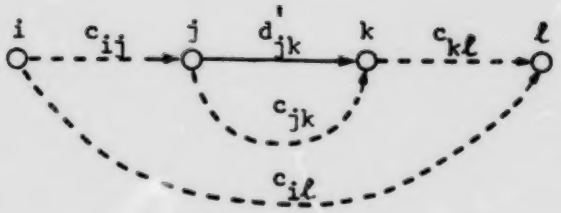
3) le plus court chemin de k à ℓ, de longueur $c_{k\ell}$.

$c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

Il est clair que $c_{i\ell}$ doit être modifié comme suit:

$$c_{i\ell} := c_{ij} + d'_{jk} + c_{k\ell} \quad .$$

2e cas



Le plus court chemin de i à ℓ ne passe pas par l'arc (j,k) .

$$c_{i\ell} < c_{ij} + d_{jk} + c_{k\ell} \ .$$

Il se peut qu'il faille tout de même modifier $c_{i\ell}$. En effet, si $c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$, un nouveau plus court chemin se trouve créé par la modification de l'arc (j,k) , qui va de i à ℓ en passant par l'arc (j,k) . Dans ce cas, il faut modifier $c_{i\ell}$ comme suit:

$$c_{i\ell} := c_{ij} + d'_{jk} + c_{k\ell} \ .$$

D'après la discussion qui précède, on peut énoncer la proposition 1 .

Proposition 1:

$c_{i\ell}$ doit être modifié $\iff c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

La proposition 1 permet de repérer toutes les paires (i,ℓ) à modifier et seulement elles. Cependant, l'utilisation de la proposition 1 implique l'énumération de $n(n-1)$ paires pour leur faire subir le test "est-ce que $c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$? ". On aura moins de paires (i,ℓ) à énumérer si l'on parvient au préalable à éliminer à l'aide d'un test simple un certain nombre d'indices i , $i \in N$, dont on est certain qu'ils ne peuvent former le premier élément d'une paire (i,ℓ) à modifier. Soit N_1 l'ensemble des indices i qui ne sont pas éliminés ainsi. On peut procéder de façon analogue pour les indices ℓ , $\ell \in N$, et former N_2 , l'ensemble des indices ℓ qui peuvent être le deuxième élément d'une paire (i,ℓ) à modifier.

$N_1 \subset N$

$$N_1 = \left\{ i \;\middle|\; \begin{array}{l} i \text{ est éligible à former le premier élément d'une paire} \\ (i,\ell) \text{ à modifier} \end{array} \right\}$$

$N_2 \subset N$

$$N_2 = \left\{ \ell \;\middle|\; \begin{array}{l} \ell \text{ peut former le deuxième élément d'une paire } (i,\ell) \text{ à} \\ \text{modifier} \end{array} \right\}$$

Précisons maintenant la nature des tests qui définissent N_1 et N_2 .

Proposition 2:

$c_{ik} \leq c_{ij} + d'_{jk} \Longrightarrow$ i ne peut former le premier élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{ik} \leq c_{ij} + d'_{jk}$

$\forall \ell$, $c_{ik} + c_{k\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$

Or $c_{i\ell} \leq c_{ik} + c_{k\ell}$ (propriété des plus courts chemins)

$\Longrightarrow \forall \ell$, $c_{i\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$

$\Longleftrightarrow$ $c_{i\ell}$ ne peut être modifié (par la proposition 1) .

Proposition 3:

$c_{j\ell} \leq d'_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut former le deuxième élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$

Fin de la preuve.

Nous définissons donc:

$$N_1 = \{ i \mid c_{ik} > c_{ij} + d'_{jk} \}$$

$$N_2 = \{ \ell \mid c_{j\ell} > d'_{jk} + c_{k\ell} \}$$

$|N_1| = n_1$ et $1 \leq n_1 \leq n$

$|N_2| = n_2$ et $1 \leq n_2 \leq n$

Les $n_1 n_2$ paires $(i,\ell) \ni i \in N_1$ et $\ell \in N_2$ sont les seules paires susceptibles d'être modifiées. La définition de N_1 et de N_2 mène directement à l'algorithme D (D pour diminution de la longueur d'un arc).

L'algorithme D

(Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changements dans C .)

d1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

d2) $\forall i \in N$, vérifier si $c_{ik} > c_{ij} + d'_{jk}$.
Si oui, $N_1 := N_1 \cup \{i\}$ et $c_{ik} := c_{ij} + d'_{jk}$.

d3) $\forall \ell \in N$, vérifier si $c_{j\ell} > d'_{jk} + c_{k\ell}$.
Si oui, $N_2 := N_2 \cup \{\ell\}$ et $c_{j\ell} := d'_{jk} + c_{k\ell}$.

d4) $\forall i \in N_1$ et $\forall \ell \in N_2$, vérifier si $c_{i\ell} > c_{ij} + c_{j\ell}$.
(à noter que depuis l'étape d3, $c_{j\ell} = d'_{jk} + c_{k\ell}$)
Si oui, $c_{i\ell} := c_{ij} + c_{j\ell}$.

Cet algorithme est essentiellement le même que celui proposé par Murchland (1970, page 39).

Nombre d'opérations élémentaires

L'utilisation de l'algorithme de Floyd-Murchland pour obtenir la nouvelle matrice C exige de l'ordre de $2n^3$ opérations élémentaires (n^3 additions, n^3 comparaisons). L'algorithme D ne demande que $2(2n + n_1 n_2)$ opérations élémentaires:

$2n$ à l'étape d2 (n additions et n comparaisons)

$+ 2n$ à l'étape d3

$+ 2n_1 n_2$ à l'étape d4 .

Dans un grand réseau, n_1 et n_2 devraient être considérablement plus petits que n . Le contraire signifierait que tous les plus courts chemins ont tendance à passer par le même arc (j,k) .

$$n_1 << n \qquad \text{et} \qquad n_2 << n \ .$$

Afin d'obtenir un estimé grossier du rapport du temps de calcul pour l'algorithme D au temps de calcul pour l'algorithme de Floyd-Murchland, considérons le rapport R du nombre d'opérations élémentaires exigées par l'algorithme D au nombre d'opérations élémentaires exigées par l'algorithme de Floyd-Murchland. Pour n assez grand, on a:

$$R = \frac{2(2n + n_1 n_2)}{2n^3} = \frac{2}{n^2} + \frac{n_1 n_2}{n^3} \cong \frac{n_1 n_2}{n^3} << \frac{n^2}{n^3} = \frac{1}{n} \ .$$

Résultats expérimentaux

Pour recueillir de l'information sur le comportement numérique de l'algorithme D, nous avons résolu 47 problèmes. Ces problèmes sont constitués à partir de quatre réseaux; pour chacun de ces réseaux, un certain nombre (que nous appellerons NPROB) de problèmes a été obtenu en modifiant (diminuant) successivement la longueur de NPROB arcs, chaque modification donnant lieu à un problème. Nous ne jugeons pas opportun de donner ici la description complète de toutes ces données numériques, mais nous les gardons disponibles pour toute demande éventuelle de consultation. Et nous nous intéressons ici aux résultats numériques obtenus.

Pour chacun des problèmes traités, nous avons recueilli les valeurs de n_1 et n_2, de NMOD (i.e. le nombre d'éléments de la matrice C effectivement modifiés) et de T (i.e. le temps de calcul en secondes pour l'algorithme D).

On trouvera les résultats numériques dans le tableau 1, où nous avons dû les présenter de façon condensée (sous forme de moyennes) pour éviter de provoquer une avalanche de valeurs numériques à cause du grand nombre de problèmes traités (47). Dans le tableau 1, lorsqu'on lit la valeur de n_1 pour un réseau donné, il s'agit en fait de la valeur moyenne de n_1 pour les NPROB problèmes rattachés à ce réseau. De même pour n_2, NMOD et T. Pour fins de comparaison, les mêmes problèmes ont été traités par l'algorithme de Floyd-Murchland et le tableau 1 donne aussi le temps de calcul moyen en secondes pour cet algorithme (T_{floyd}).

Dans le tableau 1, il est particulièrement intéressant de noter comment le rapport T/T_{floyd} a tendance à décroître rapidement quand n croît, ce qui correspond grosso modo à l'estimé que nous avons fait précédemment en considérant le rapport R . Cependant, il ne s'agit là que d'une tendance, car, pour un problème donné, les valeurs de n_1 et de n_2 (et donc de T) dépendent non seulement de n , mais aussi de la structure du réseau et de la position stratégique de l'arc modifié.

	n	NPROB	n_1	n_2	NMOD	T	T_{floyd}	T/T_{floyd}
réseau # 1	22	10	9	6	22	.0013	0.07	1.9%
réseau # 2	35	10	15	14	95	.0022	0.29	0.8%
réseau # 3	50	15	29	20	157	.0080	0.82	1.0%
réseau # 4	70	12	34	28	266	.0090	2.22	0.4%

TABLEAU 1

2.1.2 Augmentation de la longueur d'un arc

Pour les fins de la discussion qui suit, excluons le cas où $c_{jk} < d_{jk}$, car alors C reste inchangée. On a donc:

$$0 < c_{jk} = d_{jk} < d'_{jk} \leq M .$$

Proposition 4:

$c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \Longrightarrow$ on ne doit pas modifier $c_{i\ell}$.

Preuve:

$c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \Longrightarrow$ le plus court chemin de i à ℓ ne passe pas par l'arc (j,k)

$\Longrightarrow$ il ne faut pas modifier $c_{i\ell}$

Fin de la preuve.

On ne conservera donc que des paires (i,ℓ) pour lesquelles l'égalité $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ a lieu, et on rejettera celles pour lesquelles l'égalité n'a pas lieu. Cependant, il est important de noter que même si l'égalité a lieu, il se peut qu'on n'ait pas à modifier $c_{i\ell}$. Cela se produit lorsqu'il existe plus d'un chemin de longueur $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ pour aller de i à ℓ. Dans ce cas, en allongeant l'arc (j,k), on allonge aussi le plus court chemin passant par (j,k) et allant de i à ℓ, mais il existe un autre chemin de longueur $c_{i\ell}$, qui, lui, n'est pas modifié.

Lorsqu'on diminue la longueur d'un arc, le plus court chemin entre deux sommets quelconques peut tout au plus diminuer en longueur: son

parcours ne change pas. Mais, si l'on allonge un arc, le parcours du plus court chemin entre deux sommets peut changer complètement. C'est pourquoi il faut faire un calcul de plus court chemin pour toute paire (i,ℓ) susceptible d'être modifiée, c'est-à-dire pour toute paire (i,ℓ) $\ni c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$.

Proposition 5:

$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow$ i ne peut former le premier élément d'une paire (i,ℓ) à modifier .

Preuve:

$$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow c_{ik} < c_{ij} + d_{jk}$$

$$\Longrightarrow \quad \forall \ell \ , \ c_{ik} + c_{k\ell} < c_{ij} + d_{jk} + c_{k\ell}$$

Or $c_{i\ell} \leq c_{ik} + c_{k\ell}$ (propriété des plus courts chemins) .

Donc $\forall \ell \ , \ c_{i\ell} < c_{ij} + d_{jk} + c_{k\ell}$

et, par la proposition 4, $c_{i\ell}$ ne peut être modifié.

Proposition 6:

$c_{j\ell} \neq d_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut former le deuxième élément d'une paire (i,ℓ) à modifier .

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} < c_{ij} + d_{jk} + c_{k\ell}$.

Une application judicieuse des propositions 4, 5 et 6 mène directement à l'algorithme A (A pour augmentation). Cette fois-ci, on

définit:

$$N_1 = \{i \mid c_{ik} = c_{ij} + d_{jk}\}$$

$$N_2 = \{\ell \mid c_{j\ell} = d_{jk} + c_{k\ell}\}$$

Il est important de remarquer que

$$(i,\ell) \in N_1 \times N_2 \not\Longrightarrow c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} .$$

Avant d'entreprendre le calcul du plus court chemin pour une paire $(i,\ell) \in N_1 \times N_2$, on vérifiera si cette paire est susceptible d'être modifiée, c'est-à-dire si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$ (cf. proposition 4).

L'algorithme A

(Si $c_{jk} < d_{jk}$, la matrice C n'est pas modifiée.)

a1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

a2) $\forall i \in N$, si $c_{ik} = c_{ij} + d_{jk}$, $N_1 := N_1 \cup \{i\}$.

a3) $\forall \ell \in N$, si $c_{j\ell} = d_{jk} + c_{k\ell}$, $N_2 := N_2 \cup \{\ell\}$.

a4) $\forall$ paire $(i,\ell) \in N_1 \times N_2$, si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$, on conserve la paire (i,ℓ) et $c_{i\ell} := d_{i\ell}$.

(remarque: $d_{i\ell}$ est la plus petite valeur que l'on puisse attribuer a priori à $c_{i\ell}$, dont la véritable valeur sera calculée à l'étape a5.)

a5) Appliquer l'algorithme de Floyd-Murchland à toutes les paires (i,ℓ) conservées à l'étape a4, i.e.

pour $m := 1$ jusqu'à n

et pour chaque paire (i,ℓ) conservée à l'étape a4

si $c_{im} + c_{m\ell} < c_{i\ell}$ alors $c_{i\ell} := c_{im} + c_{m\ell}$.

Cet algorithme diffère essentiellement de celui proposé par Murchland (1970 , page 43): l'étape a4 n'existe pas dans l'algorithme de Murchland. En ajoutant l'étape a4, on se trouve à éviter de faire l'onéreux calcul du plus court chemin pour toute paire $(i,\ell) \in N_1 \times N_2$ pour laquelle $c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell}$. On diminue ainsi le temps de calcul, ainsi que le confirment les résultats numériques que nous avons recueillis (cf. tableau 2).

Nombre d'opérations élémentaires

Supposons qu'à l'étape a4, on conserve n_{12} paires $(i,\ell) \in N_1 \times N_2$.

$$n_{12} < n_1 n_2$$

Alors le nombre d'opérations élémentaires pour l'algorithme A s'évalue comme suit:

	nombre d'additions	nombre de comparaisons
à l'étape a2,	n	n
à l'étape a3,	n	n
à l'étape a4,	$n_1 n_2 + n_1$ (ou $n_1 n_2 + n_2$)	$n_1 n_2$
à l'étape a5,	$n_{12} n$	$n_{12} n$
au total:	$2n + n_1 + n_1 n_2 + n_{12} n$	$2n + n_1 n_2 + n_{12} n$

Négligeant le terme n_1 qui a peu d'importance par rapport aux autres, on obtient:

$$\begin{aligned}
\text{nombre d'opérations élémentaires} &= 2(2n + n_1n_2 + n_{12}n) \\
&< 2(2n + n_1n_2 + n_1n_2n) \\
&= 2((2 + n_1n_2)n + n_1n_2) \\
&\cong 2(n_1n_2n + n_1n_2) \\
&= 2(n_1n_2(n + 1)) \\
&\cong 2n_1n_2n \quad , \text{ pour } n \text{ assez grand} \\
&<< 2n^3 \text{ , puisque } n_1 << n \text{ et } n_2 << n .
\end{aligned}$$

$$R \cong \frac{2n_1n_2n}{2n^3} = \frac{n_1n_2}{n^2} << 1$$

Le cas de l'allongement d'un arc se montre plus lourd à traiter que le cas du raccourcissement. Dans ce dernier cas, s'il faut modifier $c_{i\ell}$, il suffit de changer sa valeur ($c_{i\ell} := c_{ij} + d_{jk} + c_{k\ell}$). Dans le cas de l'allongement, s'il faut modifier $c_{i\ell}$, il faut faire un calcul de plus court chemin (étape a5). On ne peut espérer réduire le nombre d'opérations élémentaires qu'exige l'étape a5, puisque l'algorithme de Floyd-Murchland utilisé effectue ce calcul en un nombre minimum d'opérations (Murchland (1970), page 13, théorème 2).

Résultats expérimentaux

Les résultats numériques pour l'algorithme A apparaissent dans le tableau 2. Ce tableau s'interprète de la même façon que le tableau 1. On y trouve cependant de nouvelles valeurs, n_{12} et T' . T' représente le temps de calcul en secondes pour l'algorithme A sans l'étape a4. T' est donc le temps de calcul qu'on obtient si l'on effectue un calcul de plus court chemin pour toutes les paires $(i,\ell) \in N_1 \times N_2$, ce qui correspond à l'algorithme proposé par Murchland (1970, page 43). Dans le tableau 2, on remarquera que $n_{12} < n_1 n_2$ et donc que $(n_1 n_2 - n_{12})$ représente un nombre appréciable de paires (i,ℓ) pour lesquelles le calcul du plus court chemin est supprimé à l'aide de l'étape a4; cela se reflète dans le fait que $T < T'$. Finalement, on remarquera qu'en utilisant l'algorithme A plutôt que l'algorithme de Floyd-Murchland, on réalise une nette économie en temps de calcul $(T < T_{floyd})$. De plus, le rapport T/T_{floyd} suit une tendance à la décroissance quand n croît (exception faite pour le réseau # 3, qui engendre une valeur de T anormalement élevée).

	n	NPROB	n_1	n_2	n_{12}	NMOD	T	T'	T_{floyd}	T/T_{floyd}
réseau # 1	22	10	11	7	38	36	.010	.022	.07	14%
réseau # 2	35	10	18	22	73	71	.030	.142	.29	10%
réseau # 3	50	15	27	28	206	168	.113	.493	.82	14%
réseau # 4	70	12	26	30	162	158	.121	.412	2.22	5%

TABLEAU 2

2.2 Le cas symétrique

Le problème à traiter reste le même que dans le cas orienté, sauf pour la symétrie (le réseau est maintenant symétrique: $d_{ij} = d_{ji}$, $\forall i$ et $\forall j$) et pour ses conséquences immédiates (par exemple, $c_{ij} = c_{ji}$, $\forall i$ et $\forall j$) . Dans le cas symétrique, on ne parle pas d'arcs, mais d'arêtes, et on ne parle pas de chemins, mais de chaînes. Pour tout éclaircissement sur cette terminologie, nous renvoyons le lecteur au livre de Berge (1967).

Nous proposons ici deux algorithmes pour calculer la nouvelle matrice des longueurs des plus courtes chaînes après modification d'une seule arête (j,k) . Le premier algorithme s'applique dans le cas d'une diminution de la longueur de l'arête (j,k) ; le second, dans le cas d'une augmentation. Pour obtenir ces algorithmes, nous avons procédé de la même façon que dans le cas orienté, en faisant les transpositions dictées par la symétrie. C'est pourquoi nous serons brefs dans l'exposé qui suit.

A notre connaissance, seuls Boyce *et al.* (1973) ont proposé des algorithmes pour le problème étudié ici (cas symétrique). Nous mentionnerons au passage en quoi leurs algorithmes ressemblent aux nôtres ou en diffèrent.

2.2.1 Diminution de la longueur d'une arête

Soit donc (j,k) l'arête modifiée et d'_{jk} , sa nouvelle longueur. Si $c_{jk} \leq d'_{jk}$, il n'y a pas de changement dans C . On supposera donc

que

$$0 < d'_{jk} < c_{jk} \leq d_{jk} \leq M .$$

Posons P = l'ensemble des paires (i,ℓ) à modifier

$$E = \{ (i,\ell) \mid c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \}$$

$$E^S = \{ (i,\ell) \mid (\ell,i) \in E \} = \text{le symétrique de } E .$$

Proposition 7: $P = E \cup E^S$.

Preuve: Soit $(i,\ell) \in P$.

Il faut donc modifier $c_{i\ell}$ et cela implique qu'on peut construire une plus courte chaîne qui emprunte l'arête (j,k) et qui est plus courte que $c_{i\ell}$. Puisque l'arête (j,k) ne comporte aucune orientation, cette plus courte chaîne est formée d'une des deux façons suivantes:

1) elle emprunte la plus courte chaîne entre i et j ,
l'arête (j,k) entre j et k ,
et la plus courte chaîne entre k et ℓ .
Dans ce cas, $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} > c_{ij} + d'_{jk} + c_{k\ell}$.

2) elle emprunte la plus courte chaîne entre i et k ,
l'arête (j,k) entre k et j ,
et la plus courte chaîne entre j et ℓ .
Dans ce cas, $c_{i\ell} = c_{ik} + d_{jk} + c_{j\ell} > c_{ik} + d'_{jk} + c_{j\ell}$.

On a donc:

$$(i,\ell) \in P \Longrightarrow \begin{cases} c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \\ \underline{\text{ou}} \quad c_{i\ell} > c_{ik} + d'_{jk} + c_{j\ell} \end{cases}$$

A cause de la symétrie,

$$c_{i\ell} > c_{ik} + d'_{jk} + c_{j\ell} \Longleftrightarrow c_{\ell i} > c_{\ell j} + d'_{jk} + c_{ki}$$

Et on peut réécrire:

$$(i,\ell) \in P \Longrightarrow (i,\ell) \in E \ \underline{\text{ou}} \ (\ell,i) \in E \Longleftrightarrow (i,\ell) \in E \cup E^s$$

Donc $P \subseteq E \cup E^s$.

Par ailleurs, $E \cup E^s \subseteq P$.

En effet, $(i,\ell) \in E \Longleftrightarrow c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell}$

$\Longrightarrow c_{i\ell}$ à modifier

$\Longleftrightarrow (i,\ell) \in P$

De même, $(i,\ell) \in E^s \Longleftrightarrow (\ell,i) \in E$

$\Longleftrightarrow c_{\ell i} > c_{\ell j} + d'_{jk} + c_{ki}$

$\Longrightarrow c_{\ell i}$ à modifier

$\Longrightarrow c_{i\ell}$ à modifier

(puisqu'à cause de la symétrie, on ne peut modifier $c_{\ell i}$ sans modifier $c_{i\ell}$)

$\Longleftrightarrow (i,\ell) \in P$

Corollaire 7a: Pour connaître l'ensemble des paires (i,ℓ) à modifier, il suffit de considérer l'ensemble

$$E = \{ (i,\ell) \mid c_{i\ell} > c_{ij} + d'_{jk} + c_{k\ell} \} ,$$

puisque $c_{i\ell}$ et $c_{\ell i}$ doivent être toujours modifiés simultanément.

Proposition 8:

$c_{ik} \leq c_{ij} + d'_{jk} \Longrightarrow$ i ne peut constituer le premier élément d'une paire (i,ℓ) à modifier.

Preuve: Comme dans le cas orienté (voir la preuve de la proposition 2), on a:

$$\forall \ell \ , \ c_{i\ell} \leq c_{ik} + c_{k\ell} \leq c_{ij} + d'_{jk} + c_{k\ell} \ .$$

Et par le corollaire 7a, on n'a pas à considérer l'indice i .

Proposition 9:

$c_{j\ell} \leq d'_{jk} + c_{k\ell} \Longrightarrow \ell$ ne peut constituer le deuxième élément d'une paire (i,ℓ) à modifier.

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} \leq c_{ij} + d'_{jk} + c_{k\ell}$.

L'utilisation judicieuse du corollaire 7a et des propositions 8 et 9 nous donne l'algorithme DS (diminution, symétrique).

L'algorithme DS

(Si $c_{jk} \leq d'_{jk}$, la matrice C reste inchangée.)

ds1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

ds2) $\forall\, i \in N$, si $c_{ik} > c_{ij} + d'_{jk}$,

$$N_1 := N_1 \cup \{i\} , \quad c_{ik} := c_{ij} + d'_{jk} \text{ et } c_{ki} := c_{ik} .$$

ds3) $\forall\, \ell \in N$, si $c_{j\ell} > d'_{jk} + c_{k\ell}$,

$$N_2 := N_2 \cup \{\ell\} , \quad c_{j\ell} := d'_{jk} + c_{k\ell} \text{ et } c_{\ell j} := c_{j\ell} .$$

ds4) $\forall\, (i,\ell) \in N_1 \times N_2$, si $c_{i\ell} > c_{ij} + c_{j\ell}$, $c_{i\ell} := c_{ij} + c_{j\ell}$

$$\text{et } c_{\ell i} := c_{i\ell} .$$

Hormis quelques légères différences au niveau des détails, l'algorithme DS et l'algorithme "link addition" proposé par Boyce *et al.* (1973, p. 524) sont essentiellement le même algorithme.

Nombre d'opérations élémentaires

Comme le montre une simple comparaison des algorithmes D et DS, l'algorithme DS exige le même nombre d'opérations élémentaires que l'algorithme D, soit $2(2n + n_1 n_2)$. Cependant, si l'on adapte l'algorithme de Floyd-Murchland au cas symétrique, le nombre d'opérations élémentaires exigées par ce dernier algorithme n'est plus que de l'ordre de n^3 (et

non plus $2n^3$, puisqu'il suffit de calculer c_{ij} pour la moitié seulement des paires de sommets, à cause de la symétrie).

Résultats expérimentaux

Le tableau 3 présente les résultats numériques que nous avons obtenus pour l'algorithme DS. Il se lit comme le tableau 1. La tendance de T/T_{floyd} à décroître quand n croît s'y manifeste clairement.

	n	NPROB	n_1	n_2	NMOD	T	T_{floyd}	T/T_{floyd}
réseau # 5	20	10	4	3	13	.0007	.03	2.3%
réseau # 6	35	10	9	11	48	.0023	.15	1.5%
réseau # 7	50	10	21	9	103	.0040	.44	0.9%
réseau # 8	70	10	25	16	204	.0080	1.19	0.7%

TABLEAU 3

2.2.2 Augmentation de la longueur d'une arête

Si $c_{jk} < d_{jk}$, C reste inchangée. On supposera donc que

$$0 < c_{jk} = d_{jk} < d'_{jk} \leq M.$$

Proposition 10:

$$\left.\begin{array}{l} c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \\ \text{et } c_{i\ell} \neq c_{ik} + d_{kj} + c_{j\ell} \end{array}\right\} \Longrightarrow \text{il ne faut pas modifier } c_{i\ell}.$$

Preuve:

$$\left.\begin{array}{l} c_{i\ell} \neq c_{ij} + d_{jk} + c_{k\ell} \\ \underline{\text{et}}\ \ c_{i\ell} \neq c_{ik} + d_{kj} + c_{j\ell} \end{array}\right\} \Longrightarrow \begin{cases} \text{la plus courte chaîne entre } i \text{ et } \ell \\ \text{n'emprunte pas l'arête } (j,k) \end{cases}$$

$\Longrightarrow$ il ne faut pas modifier $c_{i\ell}$

Fin de la preuve.

Corollaire 10a: Pour identifier les paires (i,ℓ) susceptibles d'être modifiées, il suffit de considérer l'ensemble

$$E = \{(i,\ell) \mid c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}\} \ .$$

Preuve: Par la proposition 10, on a:

$$c_{i\ell} \text{ susceptible d'être modifié} \Longrightarrow \begin{cases} c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell} \\ \underline{\text{ou}}\ \ c_{i\ell} = c_{ik} + d_{kj} + c_{j\ell} \end{cases}$$

Si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$, alors $(i,\ell) \in E$.

Si $c_{i\ell} = c_{ik} + d_{kj} + c_{j\ell}$, alors $c_{\ell i} = c_{\ell j} + d_{jk} + c_{ki}$

et $(\ell,i) \in E$.

Proposition 11:

$$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow \begin{cases} i \text{ n'a pas à être considéré comme premier élément} \\ \text{d'une paire } (i,\ell) \text{ susceptible d'être modifiée.} \end{cases}$$

Preuve:

$c_{ik} \neq c_{ij} + d_{jk} \Longrightarrow \forall \ell\ ,\ c_{i\ell} \leq c_{ik} + c_{k\ell} < c_{ij} + d_{jk} + c_{k\ell}$

et par le corollaire 10a, on n'a pas à considérer i .

Proposition 12:

$c_{j\ell} \neq d_{jk} + c_{k\ell} \Longrightarrow$ ℓ n'a pas à être considéré comme deuxième élément d'une paire (i,ℓ) susceptible d'être modifiée.

Preuve: $c_{i\ell} \leq c_{ij} + c_{j\ell} < c_{ij} + d_{jk} + c_{k\ell}$.

Le corollaire 10a et les propositions 11 et 12 mènent directement à l'algorithme AS (augmentation, symétrique).

L'algorithme AS

(Si $c_{jk} < d_{jk}$, C reste inchangée.)

as1) Initialisation: $N_1 := \emptyset$ et $N_2 := \emptyset$.

as2) $\forall\, i \in N$, si $c_{ik} = c_{ij} + d_{jk}$, $N_1 := N_1 \cup \{i\}$.

as3) $\forall\, \ell \in N$, si $c_{j\ell} = d_{jk} + c_{k\ell}$, $N_2 := N_2 \cup \{\ell\}$.

as4) $\forall\, (i,\ell) \in N_1 \times N_2$, si $c_{i\ell} = c_{ij} + d_{jk} + c_{k\ell}$,

on conserve la paire (i,ℓ) et $c_{i\ell} := d_{i\ell}$

$c_{\ell i} := d_{i\ell}$.

as5) Appliquer l'algorithme de Floyd-Murchland à toutes les paires (i,ℓ) conservées à l'étape as4, i.e.

pour $m := 1$ jusqu'à n

et pour chaque paire (i,ℓ) conservée

si $c_{im} + c_{m\ell} < c_{i\ell}$ alors $c_{i\ell} := c_{im} + c_{m\ell}$

et $c_{\ell i} := c_{i\ell}$.

Boyce *et al.* (1973, page 524) ont proposé un algorithme, qu'ils appellent "link deletion", pour accomplir le même travail que l'algorithme AS. Hormis le fait qu'il comporte deux légères erreurs qui pourraient bien n'être que des erreurs typographiques* , leur algorithme diffère essentiellement du nôtre en ce qu'il ne comporte pas l'étape as4. Or cette étape permet de réduire le temps de calcul, ainsi que le montre l'expérimentation (cf. tableau 4).

Nombre d'opérations élémentaires

Une simple comparaison entre l'algorithme A et l'algorithme AS nous indique qu'ils comportent exactement le même nombre d'opérations élémentaires, soit $2(2n + n_1 n_2 + n_{12} n)$ additions et comparaisons.

*Note: à l'étape 1 de l'algorithme "link deletion", il faudrait lire "$c_{jk} < d_{jk}$" au lieu de "$c_{jk} \leq d_{jk}$", et à l'étape 2(a), il faudrait lire "$c_{ik} = c_{ij} + d_{jk}$" et non pas "$c_{ij} = c_{ij} + d_{jk}$".

Résultats expérimentaux

Le tableau 4 présente les résultats numériques que nous avons recueillis avec l'algorithme AS. Il s'interprète exactement comme le tableau 2. On y observera encore une fois que le rapport T/T_{floyd} décroît quand n croît.

	n	NPROB	n_1	n_2	n_{12}	NMOD	T	T'	T_{floyd}	T/T_{floyd}
réseau # 5	22	10	3	5	13	13	.004	.005	.03	13%
réseau # 6	35	10	8	12	47	45	.019	.035	.15	13%
réseau # 7	50	10	30	5	81	79	.045	.077	.44	10%
réseau # 8	70	10	34	5	63	62	.049	.102	1.19	4%

TABLEAU 4

L'algorithme AS sera utilisé au chapitre suivant comme une des composantes d'un algorithme qui résoud le problème du choix optimal d'un réseau. Il servira à calculer les nouvelles longueurs des plus courtes chaînes chaque fois qu'on enlèvera une arête dans un réseau symétrique. Il permettra de réaliser une économie de temps de calcul considérable, là où les calculs se faisaient auparavant par l'algorithme de Floyd-Murchland.

CHAPITRE III

RECHERCHE D'UN ALGORITHME EFFICACE POUR LE PROBLEME DU CHOIX OPTIMAL D'UN RESEAU

3.1 Introduction

Dans la résolution du problème du choix optimal d'un réseau, une partie importante du temps de calcul se passe à faire des calculs de plus courts chemins (ou de plus courtes chaînes). En particulier, dans un algorithme proposé par Hoang (1971) pour résoudre le problème du choix optimal d'un réseau, on retrouve souvent la situation suivante: connaissant la matrice des longueurs (des plus courtes chaînes) d'un réseau symétrique et modifiant la longueur d'une seule arête du réseau, il faut calculer la nouvelle matrice des longueurs. Or les algorithmes que nous avons mis au point dans le chapitre précédent sont tout-à-fait adaptés à ce genre de calcul. Nous avons donc voulu les implanter dans l'algorithme de Hoang (1971), dans le but de réduire le temps de calcul. En cours de travail, nous avons apporté de nouvelles idées, qui allaient permettre d'obtenir un algorithme encore plus efficace.

Dans le présent chapitre, nous commençons par présenter l'algorithme initial (celui que Hoang (1971) appelle "branch search"). Nous le faisons dans une notation et selon un point de vue qui diffèrent de ceux de Hoang, mais qui nous paraissent plus appropriés pour expliquer les modifications que nous ferons subir à l'algorithme initial par la suite. Nous décrivons ensuite comment nous avons procédé, par modifications successives de l'algorithme initial fondées sur des résultats numériques, pour améliorer l'algorithme initial. L'algorithme résultant diffère sensiblement de l'algorithme initial et il se révèle beaucoup plus efficace.

3.2 Présentation de l'algorithme initial

3.2.1 Notation

Soit $R = (G,L)$ un réseau symétrique connexe défini par la donnée d'un graphe $G = (N,A)$ non orienté et connexe ainsi que d'une fonction L, définie sur A, qui associe à chaque arête de G une longueur positive.

$G = (N,A)$

$N = \{1,2,\ldots,n\}$ = ensemble des points ou sommets du graphe G

$A = \{a_1,a_2,\ldots,a_m\}$ = ensemble des arêtes du graphe G

$M = \{1,2,\ldots,m\}$

$L : A \longrightarrow \mathbb{R}^+$

$a_j \xmapsto{L} \ell_j$, $\ell_j > 0$, $j \in M$.

Nous employons G_p comme désignation générique d'un graphe partiel de G :

$$G_p = (N,A') \quad , \quad A' \subseteq A .$$

Un réseau partiel R_p se trouve défini par la donnée de G_p et par la restriction à A' de la fonction L :

$$R_p = (G_p, \text{ L restreinte à } A') .$$

R_p est dit <u>connexe</u> si G_p est connexe.

Une correspondance biunivoque s'établit naturellement entre l'ensemble des réseaux partiels (R_p) de R et l'ensemble des vecteurs caractéristiques (X^p) des A', $A' \subseteq A$:

$$R_p \longleftrightarrow X^p$$

$$X^p = (x_1^p, x_2^p, \ldots, x_m^p) \quad , \quad x_j^p = \begin{cases} 1, & \text{si } a_j \in A' \\ 0, & \text{sinon .} \end{cases}$$

Posons, par définition:

$$c_{ij}(X^p) = \text{longueur de la plus courte chaîne du sommet } i \text{ au sommet } j \text{ dans } R_p \quad (i,j \in N)$$

$$C(X^p) = \left(c_{ij}(X^p) \right)_{n \times n}$$

$$F(X^p) = \sum_{\substack{i,j \\ i<j}} c_{ij}(X^p) \quad .$$

Si l'arête a_k fait partie du réseau* X^p et a pour extrémités les sommets i et j, alors a_k peut s'écrire $a_k = (i,j)$ et nous

*Note: Puisque R_p et X^p se correspondent biunivoquement, nous dirons "le réseau X^p" plutôt que "le réseau partiel représenté par le vecteur X^p".

définissons:

$$f_k(X^p) = c_{ij}(X^p - e_k) - c_{ij}(X^p)$$

$$Q_k(X^p) = F(X^p - e_k) - F(X^p)$$

$$Q_k(X^p) \geq f_k(X^p)$$

$$F(X^p - e_k) = F(X^p) + Q_k(X^p) \geq F(X^p) + f_k(X^p)$$

où e_k est le $k^{\text{ième}}$ vecteur canonique à m composantes, i.e. $e_k = (0,0,\ldots,0,1,0,0,\ldots,0)$.

un seul 1 *en position* k

$F(X^p)$ constitue une mesure de la "longueur totale" du réseau X^p. $f_k(X^p)$ correspond à l'augmentation de longueur que l'on fait subir à la plus courte chaîne entre i et j lorsqu'on enlève l'arête a_k du réseau X^p ; f_k n'est donc qu'une mesure locale de l'effet résultant de la suppression de l'arête a_k. Quant à $Q_k(X^p)$, il indique l'augmentation totale de longueur dans le réseau X^p causée par la suppression de a_k ; Q_k correspond donc à une mesure globale de l'effet résultant de cette suppression. $Q_k(X^p)$ et $f_k(X^p)$ constituent donc des mesures de l'utilité de l'arête a_k dans le réseau X^p : plus $Q_k(X^p)$ — ou $f_k(X^p)$ — est grand, plus il est utile de conserver l'arête a_k dans le réseau X^p. Evidemment, Q_k est une meilleure mesure d'utilité que f_k.

Nous appelons solution n'importe quel vecteur à m composantes dont toutes les composantes sont 0 ou 1 . A toute solution correspond un réseau partiel et vice-versa. L'ensemble de toutes les solutions s'écrit

$$S = \{X = (x_1, x_2, \ldots, x_m) \mid \quad x_j = 0 \text{ ou } 1,\ j \in M\} .$$

Nous appelons solution réalisable une solution qui satisfait les contraintes du problème du choix optimal d'un réseau (défini ci-dessous, cf. section 3.2.2), et nous appelons solution optimale une solution réalisable qui minimise la fonction économique du même problème.

3.2.2 Définition du problème du choix optimal d'un réseau

Nous considérons ici le problème du choix optimal d'un réseau dans sa forme la plus simple. En particulier, nous supposons que la demande O-D est la même entre toutes les paires de sommets. Cependant, nous verrons au chapitre V comment on peut traiter une forme plus générale du même problème en augmentant très peu l'effort de calcul. Nous réintroduirons alors l'usage de la matrice de demandes O-D.

Le problème à résoudre s'énonce comme suit:
Etant donné un réseau R et un budget B , trouver un réseau partiel R_p tel que:

- R_p est connexe (contrainte a)
- $\sum_{j=1}^{m} \ell_j x_j^p \leq B$ (contrainte b)
- $F(X^p)$ est minimum.

On saisit immédiatement la nature combinatoire du problème lorsqu'on songe que l'évaluation de la fonction économique F exige la connaissance de la longueur de $n(n-1)/2$ plus courtes chaînes, que la vérification de la contrainte a peut exiger autant d'efforts que l'évaluation de la fonction économique et que l'ensemble S de toutes les solutions est de cardinalité 2^m .

3.2.3 L'arborescence associée

L'algorithme de Hoang (1971) est une P.S.E., une Procédure d'exploration par Séparation et Evaluation (suivant la terminologie établie par B. Roy (1969); en anglais, branch-and-bound method, cf. Lawler and Wood (1966)). Grosso modo, une P.S.E. cherche à séparer l'ensemble S de toutes les solutions en sous-ensembles de cardinalité de plus en plus petite, donnant lieu à des problèmes de plus en plus faciles à résoudre, la minimisation se faisant sur chacun de ces sous-ensembles plutôt que sur l'ensemble S . A partir de S , une P.S.E. génère donc, à l'aide d'une partition, un certain nombre de sous-ensembles; chacun des sous-ensembles ainsi générés sera évalué; cette EVALUATION cherche à éliminer chaque sous-ensemble généré, ou bien parce qu'il ne renferme aucune solution intéressante, ou bien parce que la meilleure solution réalisable qu'il contient surclasse (et remplace) la meilleure solution réalisable obtenue jusqu'alors; tout sous-ensemble généré et non éliminé sera, à son tour, séparé par une partition (SEPARATION). Et ainsi de suite jusqu'à ce que tous les sous-ensembles générés aient été éliminés. La meilleure solution réalisable conservée se trouve alors être la solution optimale.

La relation de descendance qui lie un sous-ensemble à ceux qu'il engendre par séparation se représente commodément par une arborescence, que nous appellerons arborescence associée (pour un exemple, voir fig. 1).

Dans l'algorithme initial (Hoang (1971)), la séparation d'un sous-ensemble de S s'effectue à l'aide d'une bi-partition. Les sous-ensembles considérés sont du type suivant:

$$E = \left\{ (x_1, x_2, \ldots, x_m) \;\middle|\; \begin{array}{l} x_j \text{ peut prendre n'importe quelle valeur (0 ou 1) si } j \in L \\ \\ x_j \text{ a une valeur fixe si } j \in M - L \end{array} \right\}$$

où L est l'ensemble des indices des variables libres

et $M - L$, l'ensemble des indices des variables fixées.

$|E| = 2^{|L|}$.

Les bi-partitions considérées sont du type suivant: étant donné un sous-ensemble E du type décrit ci-dessus, on choisit un indice $r \in L$ et on sépare E en ses deux sous-ensembles,

$$E \cap \{(x_1, x_2, \ldots, x_m) \in S \mid x_r = 0\} \quad \text{et} \quad E \cap \{(x_1, x_2, \ldots, x_m) \in S \mid x_r = 1\} .$$

Nous appelons <u>variable séparatrice</u> la variable x_r qui définit la séparation.

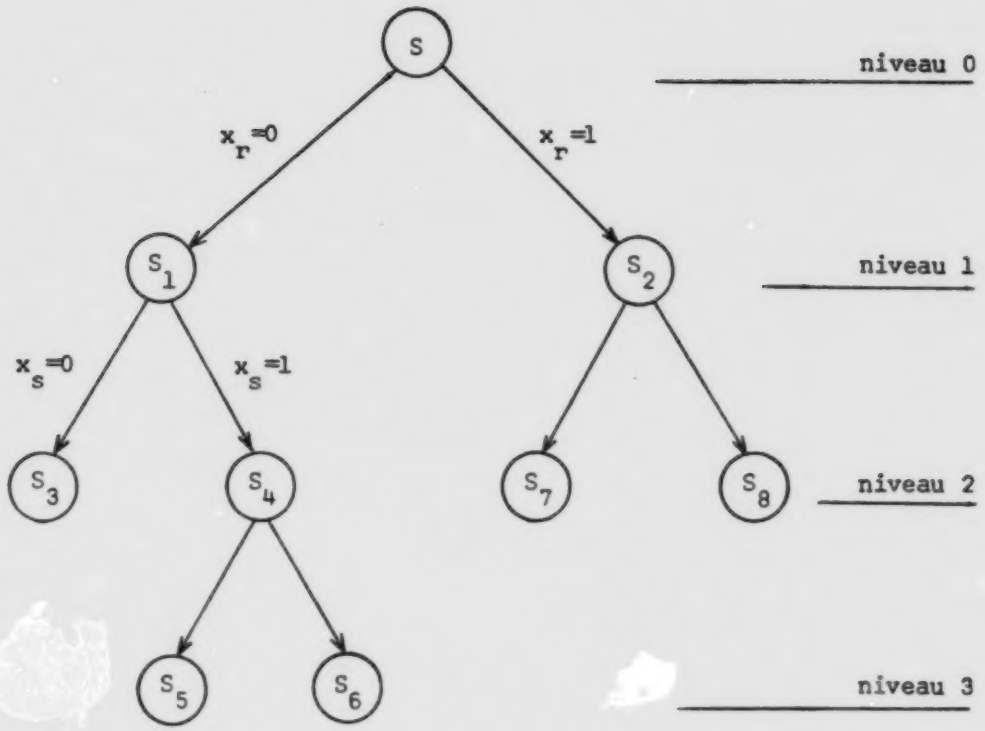


Figure 1

Exemple d'arborescence générée par l'algorithme:

L'ensemble S se trouve associé au sommet racine (à ce moment, L = M) . Puis S se trouve séparé en 2 sous-ensembles S_1 et S_2 par le choix d'un indice $r \in L$.

$S_1 = \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 0\}$ et $S_2 = \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 1\}$.

L devient M - {r} .

Par le choix d'un indice $s \in L$, x_s devient la variable séparatrice pour S_1, qu'on sépare en S_3 et S_4 .

$S_3 = \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 0, x_s = 0\}$

et $S_4 = \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 0, x_s = 1\}$.

Etc.

L'algorithme initial génère et évalue les sous-ensembles de façon séquentielle (dans la notation de Roy (1969), il s'agit d'une P.S.E.S. — S pour séquentielle). Lorsqu'on évalue un sous-ensemble E , il importe de connaître quelles sont les variables fixées dans E (avec leur valeur) ainsi que l'ordre dans lequel elles ont été fixées. A cette fin, nous utilisons une pile P définie comme suit.

Lorsqu'on évalue un sous-ensemble E , les éléments dans la pile sont les indices (précédés ou non du signe " - ") des variables fixées dans E :

$$\left|\begin{array}{ll} x_j = 0 & \text{est représenté par } -j, \\ x_j = 1 & \text{est représenté par } j \ . \end{array}\right.$$

L'indice au bas de la pile est celui de la première variable fixée et l'indice sur le dessus de la pile, celui de la dernière variable fixée. L'exemple suivant illustrera cette notation:

P = | 4 |
| -7 |
| 3 |
| -5 |

signifie qu'on a fixé d'abord $x_5 = 0$
puis $x_3 = 1$
puis $x_7 = 0$
et enfin $x_4 = 1$.

Ainsi, la pile P décrit complètement le sous-ensemble qu'on est en train d'évaluer. Nous noterons ce sous-ensemble par E^P .

$$E^P = \left\{ (x_1, x_2, \ldots, x_m) \in S \;\middle|\; \begin{array}{l} x_j = 0 \quad \text{si} \quad -j \quad \text{apparaît dans P} \\ \\ x_j = 1 \quad \text{si} \quad j \quad \text{apparaît dans P} \end{array} \right\}$$

Quant à l'ensemble des variables libres dans E^P, nous le désigne-

rons par L^P :

L^P = ensemble des indices des variables dont l'indice n'apparaît pas dans la pile P .

Dans le sous-ensemble E^P, nous distinguons la solution qui représente le réseau comptant le plus grand nombre d'arêtes et nous la notons X^P . Dorénavant, X^P ne désignera plus une solution quelconque, mais un élément bien particulier de E^P .

$$X^P = (x_1^P, x_2^P, \ldots, x_m^P) \quad , \quad \text{où} \quad x_j^P = \begin{cases} 0 \text{ , si } -j \text{ apparaît dans } P \\ 1 \text{ , autrement.} \end{cases}$$

$\forall X \in E^P$, on a : $F(X^P) \leq F(X)$.

3.2.4 La borne inférieure I^P

Lorsqu'on examine le sous-ensemble E^P , on cherche à l'éliminer de diverses façons. L'une de ces façons consiste à calculer une borne inférieure de l'ensemble $\{F(X) \mid X \in E^P$ et X réalisable$\}$, dénotée par I^P . Si $I^P \geq F(X^\circ)$, où X° représente la meilleure solution courante, on dit que la borne effectue une coupe (ou tout simplement coupe) et on peut alors éliminer le sous-ensemble E^P , car il ne contient sûrement aucune solution réalisable X meilleure que X° (i.e. telle que $F(X) < F(X^\circ)$) . Voyons comment obtenir cette borne.

Théorème I: $\forall X \in E^P$, $X = (x_1, x_2, \ldots, x_m)$, on a:

$$F(X) \geq F(X^P) + \sum_{j \in L^P} \bar{x}_j \, f_j(X^P) \, ,$$

où $\bar{x}_j = 1 - x_j$.

Le théorème est de Hoang (1971) et nous offrons la preuve suivante.

Preuve: Soit $X \in E^P$.

Si le réseau X est disconnexe, alors $F(X) = \infty$ et le théorème est nécessairement vérifié. Supposons donc X connexe.

$X \in E^P$ signifie que X est obtenu de X^P par suppression d'un certain nombre d'arêtes, disons s $(s \leq m)$. Pour simplifier, on supposera que les arêtes sont numérotées de façon que les s arêtes supprimées soient les s premières.

$$(*) \quad \begin{cases} x_j = 0 \ , \quad \text{mais} \quad x_j^P = 1 \ , & \text{pour } j \leq s \\ x_j = x_j^P \ , & \text{pour } j > s \end{cases}$$

Note: $L^P \supseteq \{1,2,\ldots,s\}$, mais $\forall j \in L^P \ni j > s$, on a: $x_j = 1$.

Supposons que le réseau Y diffère du réseau X par le seul fait qu'il contient une arête de plus que X , disons l'arête a_s .

$$(**) \quad X = Y - e_s \quad \text{i.e.} \begin{cases} y_s = 1, \ \text{mais} \ x_s = 0 \ , \\ y_j = x_j \ , \ \forall j \in M - \{s\} \ . \end{cases}$$

(***) $F(X) \geq F(Y) + f_s(Y)$, ce qui découle de la définition de $f_s(Y)$.

(*En effet, si* $a_s = (u,v)$, *alors* $c_{ij}(X) \geq c_{ij}(Y)$, $ij \neq uv$, *et par définition,* $c_{uv}(X) = c_{uv}(Y) + f_s(Y)$.)

La preuve procède par induction sur s .

Si $s = 1$, $F(X) \geq F(X^P) + f_s(X^P)$, par (***) .

Supposons le théorème vrai pour s-1 et démontrons-le pour s quelconque.

$$F(X^P) + \sum_{j \in L^P} \bar{x}_j \, f_j(X^P) = F(X^P) + \sum_{j=1}^{s-1} \bar{x}_j \, f_j(X^P) + f_s(X^P) \text{ , par (*)}$$

$$= F(X^P) + \sum_{j=1}^{s-1} \bar{y}_j \, f_j(X^P) + f_s(X^P) \text{ , par (**)}$$

$$= F(X^P) + \sum_{j \in L^P} \bar{y}_j \, f_j(X^P) + f_s(X^P) \text{ ,}$$

par (*, note) et par (**)

$$\leq F(Y) + f_s(X^P) \text{ , par l'hypothèse d'induction}$$

$$\leq F(Y) + f_s(Y) \text{ , parce que } f_s(Y) \geq f_s(X^P)$$

(en effet, lorsqu'on supprime l'arête $a_s = (u,v)$, ou bien la longueur de la plus courte chaîne de u à v ne change pas ($f_s = 0$), ou bien la plus courte chaîne de u à v est remplacée par une chaîne plus longue. Dans ce dernier cas, les possibilités de remplacement dans Y ne constituent qu'une partie des possibilités de remplacement dans X^P, et donc $f_s(Y) \geq f_s(X^P)$.)

$$\leq F(X) \text{ , par (***) .} \qquad \text{C.Q.F.D.}$$

Considérons que $X^+ = (x_1^+, x_2^+, \ldots x_m^+)$ est la solution optimale du problème (1):

$$(1)\begin{cases} \text{MIN} \quad F(X^P) + \sum\limits_{j \in L^P} \bar{x}_j \, f_j(X^P) \\ \text{en respectant les contraintes} \\ \quad (a) \quad X = (x_1, x_2, \ldots, x_m) \text{ représente un réseau connexe} \\ \quad (b) \quad \sum\limits_{j=1}^{m} \ell_j \, x_j \leq B \\ \quad (c) \quad X \in E^P \ . \end{cases}$$

Soit $\alpha = F(X^P) + \sum\limits_{j \in L^P} \bar{x}_j^+ \, f_j(X^P)$. Alors, par le théorème I, α vérifie l'inégalité $\alpha \leq \min \{F(X) \mid X \in E^P \text{ et } X \text{ réalisable}\}$ et constitue une borne inférieure de l'ensemble

$$\{F(X) \mid X \in E^P \text{ et } X \text{ réalisable}\} \ .$$

Cependant, résoudre le problème (1) s'avère coûteux en temps de calcul et d'ailleurs nullement avantageux, puisque cela exige un effort de calcul du même ordre que la recherche pure et simple de la meilleure solution réalisable dans E^P. On se contentera plutôt de calculer une borne inférieure un peu plus faible, mais tout de même utile, en relâchant les contraintes (a) et (c) du problème (1), ce qui donne lieu au problème (2).

$$
(2)\quad \begin{cases} \text{MIN} \quad F(X^P) + \sum\limits_{j \in L^P} \bar{x}_j \, f_j(X^P) \\ \text{en respectant les contraintes} \\ \text{(b)} \quad \sum\limits_{j=1}^{m} \ell_j \, x_j \leq B \\ \text{(c')} \; x_j = 0 \quad \text{si} \quad -j \quad \text{apparaît dans la pile} \quad P \\ \qquad x_j = 1 \quad \text{si} \quad j \quad \text{apparaît dans la pile} \quad P \\ \qquad 0 \leq x_j \leq 1 \, , \quad x_j \quad \text{réel, si} \quad j \in L^P \, . \end{cases}
$$

On constate que le problème (2) est tout simplement un problème de sac alpin (knapsack problem) dans lequel certaines variables peuvent prendre des valeurs fractionnaires. Ce problème s'avère très facile à résoudre. En effet, supposons que les variables ont été ordonnées (dans les calculs, on modifiera les indices) de telle façon que

$$
L^P = \{1,2,\ldots,e\} \quad , \quad \text{où} \quad e \leq m \, ,
$$

$$
\text{et que} \quad \frac{f_1(X^P)}{\ell_1} \leq \frac{f_2(X^P)}{\ell_2} \leq \ldots \leq \frac{f_e(X^P)}{\ell_e} \quad ,
$$

alors la solution optimale évidente de (2) s'écrit (cf. Dantzig (1957)):

$$X^* = (x_1^*, x_2^*, \ldots, x_m^*)$$

$$\text{où} \begin{cases} x_j^* = 0 \quad , \text{ si } j < r \\ x_j^* = 1 \quad , \text{ si } j > r \\ x_r^* = 1 - \left[\left(B - \sum_{\substack{j \in P \\ j>0}} \ell_j - \sum_{i=1}^{r-1} \ell_i \right) \Big/ \ell_r \right] , \\ \text{où } r \text{ est le plus petit entier } \ni\ 0 < r \leq e \text{ et } \sum_{i=1}^{r} \ell_i > B - \sum_{\substack{j \in P \\ j>0}} \ell_j \end{cases}$$

$$\boxed{I^P = F(X^P) + \sum_{j \in L^P} \bar{x}_j^* f_j(X^P) .}$$

3.2.5 Le critère de choix de la variable séparatrice

Hoang (1971) propose que l'indice r qui définit la variable séparatrice x_r soit justement le même indice r qu'on calcule en solutionnant le problème (2), c'est-à-dire le plus petit entier $\ni$

$$0 < r \leq e \quad \text{et} \quad \sum_{i=1}^{r} \ell_i > B - \sum_{\substack{j \in P \\ j>0}} \ell_j .$$

Comme nous serons appelés plus loin à modifier ce critère de choix de la variable séparatrice, nous introduisons dès maintenant une nouvelle notation qui permettra d'écrire d'une façon concise n'importe quel critère de choix de la variable séparatrice: dans cette notation, nous écrivons entre paranthèses 1) la mesure d'utilité par rapport à laquelle on ordonne les variables (par exemple, f_j/ℓ_j) , et 2) le rang de la va-

riable à choisir comme variable séparatrice. Dans cette notation, le critère de choix proposé par Hoang (1971) s'écrit $(f_j/\ell_j, r)$.

3.2.6 La solution initiale

Hoang (1971) propose deux méthodes pour obtenir une solution initiale (réalisable), mais sans mentionner quels sont les avantages respectifs de ces méthodes.

La première méthode, que nous appellerons méthode du graphe partiel connexe minimal (minimum spanning tree) est la plus simple. Elle consiste à obtenir le graphe partiel connexe dont la somme des longueurs des arêtes est minimale et à lui ajouter une à une les arêtes les plus courtes parmi celles qui n'ont pas encore été choisies aussi longtemps qu'on ne dépasse pas le budget.

La deuxième méthode utilise les Q_j et nous l'appellerons méthode des Q_j . Elle exige plus de calculs, mais devrait fournir une meilleure solution initiale. Elle procède comme suit:

(i) Poser $X = (1,1,\ldots,1)$.

(ii) $\forall\, j \in M \ni x_j = 1$, évaluer $Q_j(X)$.

Puis trouver $k \ni Q_k(X) = \min_{\{j \in M \mid x_j = 1\}} Q_j(X)$. Ensuite

a) si $Q_k(X) = \infty$, recommencer les calculs en employant la méthode du graphe partiel connexe minimal, car il n'est plus possible d'enlever une arête sans perdre la connexité;

b) sinon, poser $x_k = 0$. Aller en (iii) .

(iii) Si $\sum_{j=1}^{m} \ell_j x_j > B$, aller en (ii). Sinon, aller en (iv).

(iv) Si le budget le permet, tenter de redonner à une composante nulle de X la valeur 1 de façon à minimiser $F(X)$. Répéter jusqu'à ce qu'il ne soit plus possible de modifier aucune composante de X . X est alors la solution initiale cherchée.

3.2.7 Définition de l'algorithme de Hoang

INITIALISATION: On détermine d'abord une solution initiale X^o , par l'une des méthodes précitées. On pose alors $E^p = S$, c'est-à-dire que la pile P est vide, et on procède à l'évaluation de E^p .

EVALUATION: On examine le sous-ensemble E^p pour tenter de l'éliminer.

(1) On vérifie si E^p contient au moins un réseau connexe; cela revient à se demander si X^p est un réseau connexe. Si tel n'est pas le cas, E^p se trouve éliminé et on effectue un retour en arrière (voir détails ci-après).

(2) Si X^p est connexe (contrainte a), on vérifie si X^p satisfait la contrainte b du problème du choix optimal d'un réseau. Si oui, X^p est une solution réalisable et c'est, de toutes les solutions dans E^p , celle qui donne la plus petite valeur à la fonction économique F , puisque c'est elle qui comporte le plus d'arêtes. Dans ce cas, si

$(F(X^p) , \sum_{j=1}^{m} \ell_j x_j^p)$ est lexicographiquement plus petit que

$(F(X^o) , \sum_{j=1}^{m} \ell_j x_j^o)$, alors X^p remplace X^o comme meilleure

solution réalisable courante. E^P se trouve ainsi éliminé et on fait un retour en arrière.

(3) Si X^P ne respecte pas la contrainte b, on tente à nouveau d'éliminer E^P en calculant la borne inférieure I^P.
Si $I^P \geq F(X^o)$, on élimine E^P et on effectue un retour en arrière; sinon, on procède à la séparation de E^P.

SEPARATION: Le calcul de I^P a permis de déterminer un indice $r \in L^P$. On sépare E^P en 2 sous-ensembles:

$$E_o = E^P \cap \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 0\}$$

et $E^P \cap \{(x_1,x_2,\ldots,x_m) \in S \mid x_r = 1\}$.

On ajoute alors $-r$ sur le dessus de la pile P. E^P se trouve ainsi remplacé par E_o. On procède alors à l'évaluation de E^P.

RETOUR EN ARRIERE (backtracking): Effectuer un retour en arrière consiste à revenir à l'évaluation du dernier sous-ensemble généré mais non encore évalué. Lorsque tous les sous-ensembles générés ont été examinés, l'algorithme s'arrête et X^o est la solution optimale. ◆Fin.

L'algorithme n'élimine que les sous-ensembles qu'on sait ne sûrement pas contenir la solution optimale. En ce sens, il effectue une énumération partielle mais exhaustive des solutions qui conduit à la solution optimale. De plus, l'algorithme s'arrête après un nombre fini d'opérations, puisque le nombre de sous-ensembles de S est fini.

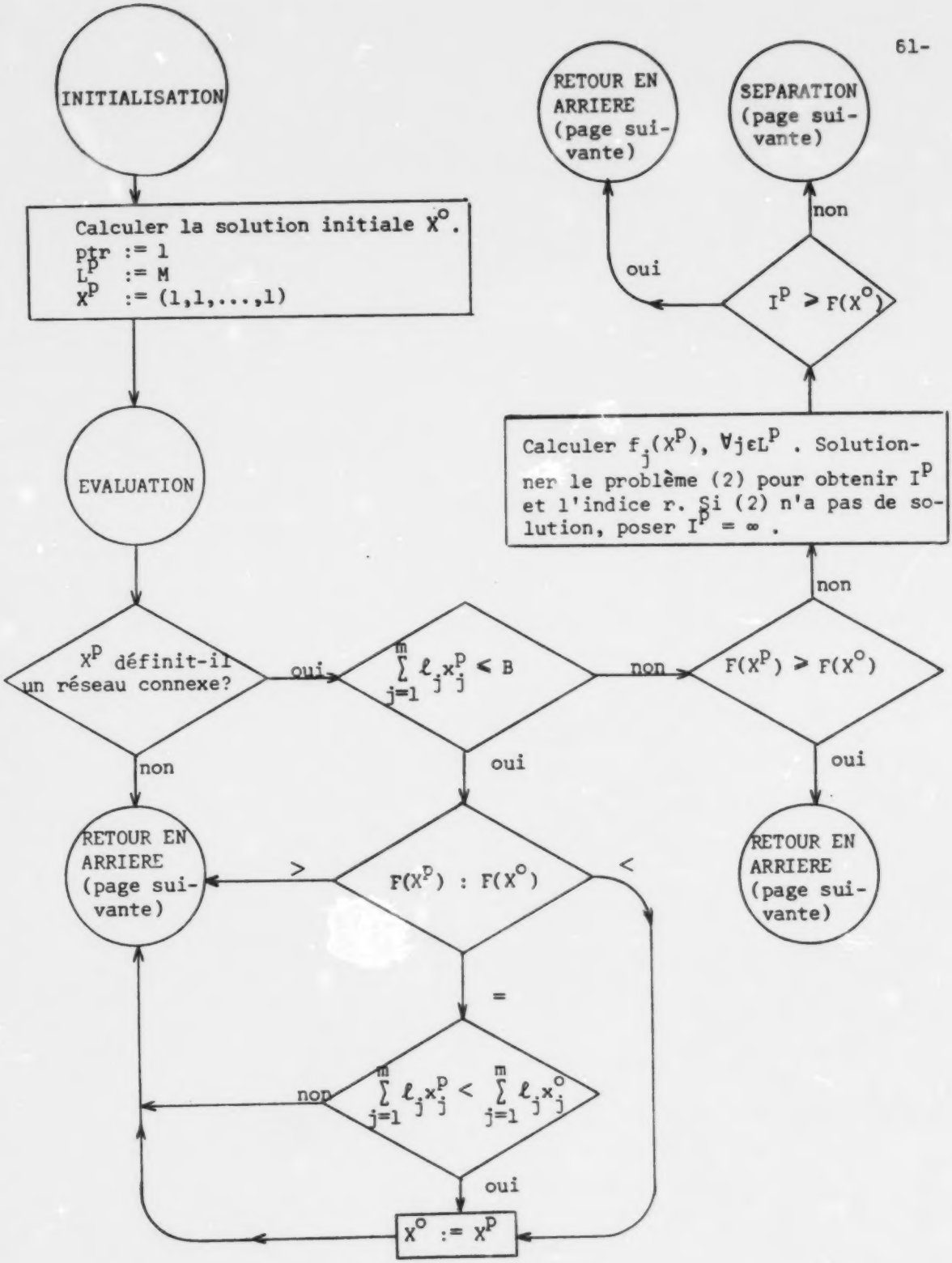
Nous allons maintenant donner l'organigramme de l'algorithme, ce qui constituera une définition formelle de l'algorithme. Dans cet organigramme, la pile P est représentée par un vecteur P à m composantes

ainsi que par un pointeur ptr , qui indique toujours le sommet de la pile. Les m composantes du vecteur P sont notées P(1) , P(2),..., P(m) et les composantes P(j) , ptr $\leq$ j $\leq$ m , doivent être considérées comme vides.

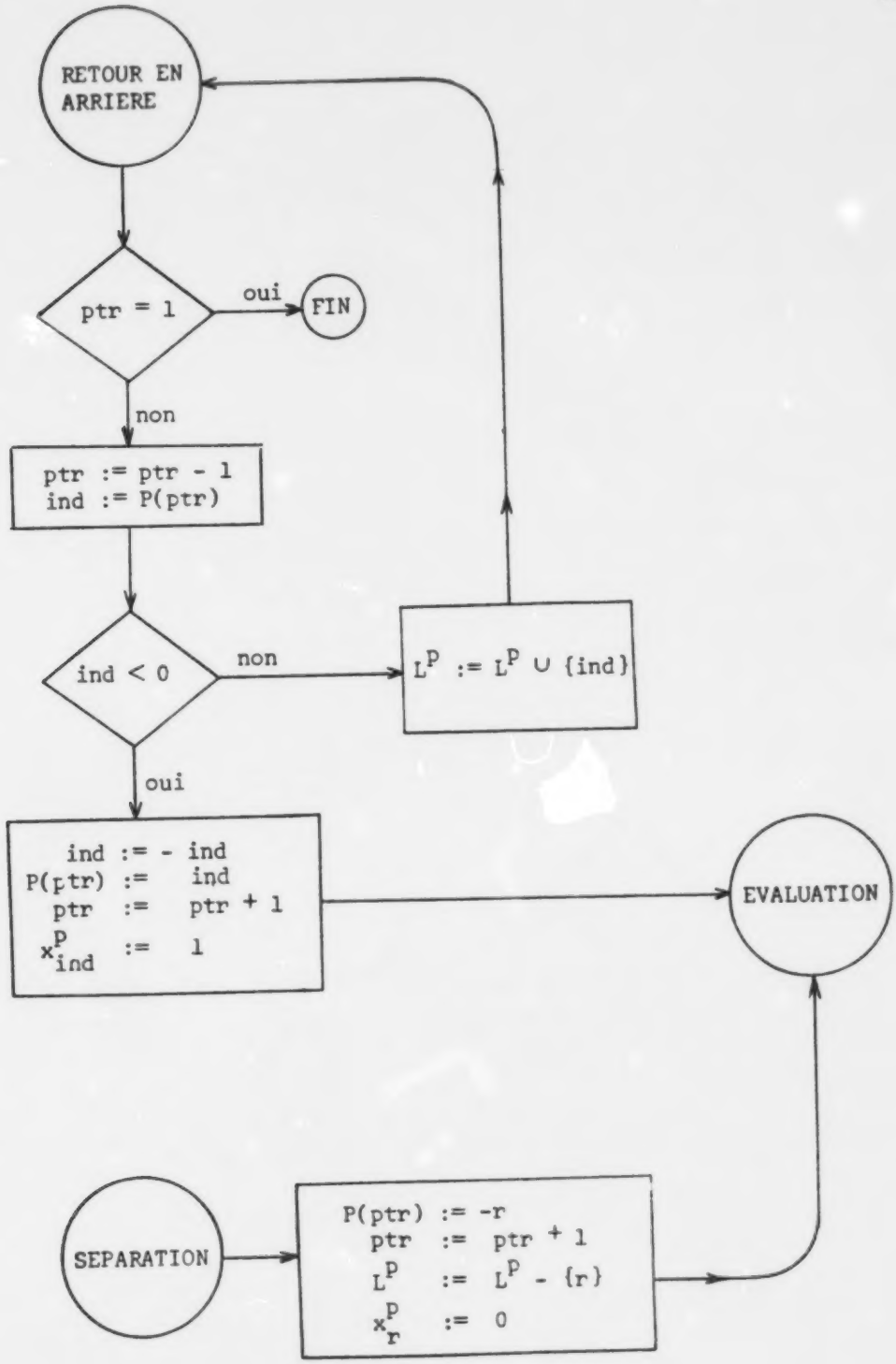
Exemple: dans cette notation, l'exemple de la page 50 s'écrit:

P = (-5, 3, -7, 4, -, -, ..., -).

ptr = 5



ORGANIGRAMME DE L'ALGORITHME DE HOANG



ORGANIGRAMME (suite)

3.2.8 Quelques remarques d'ordre pratique sur l'implantation de l'algorithme

La définition de l'algorithme est peu explicite sur la façon de mener certains calculs. En utilisant quelques astuces (et nous pensons ici à la programmation), on pourra épargner beaucoup de temps de calcul.

Remarque 1: Une façon économique de déterminer si le réseau X^p est connexe consiste à lui associer un vecteur DEG à n composantes, qu'on tiendra à jour tout au long des calculs.

DEG(i) = le degré du sommet i dans le réseau partiel X^p, $i \in N$. Le degré d'un sommet i est défini comme le nombre d'arêtes qui ont i comme extrémité. Initialement, on pose DEG(i) égal au degré du sommet i dans le réseau R, $\forall i \in N$. Ensuite, chaque fois qu'on supprime une arête $a_k = (i,j)$, c'est-à-dire chaque fois que la valeur de x_k^p passe de 1 à 0, on soustrait 1 à DEG(i) et DEG(j). Inversement, DEG(i) et DEG(j) augmentent de 1 si la valeur de x_k^p passe de 0 à 1. Si on procède ainsi, il est clair que la proposition suivante est vraie.

Proposition: $\exists i_o \in N \ni DEG(i_o)=0 \Longrightarrow$ le réseau X^p est disconnexe.

Cependant la contraposée de cette proposition n'est pas vraie en général. Pour identifier que X^p est disconnexe, il suffira souvent de tester si le vecteur DEG comporte une composante nulle. Cependant, si aucune composante de DEG n'est nulle, cela n'implique pas la connexité du réseau X^p et il faudra en vérifier au long la connexité (tous les éléments de la matrice $C(X^p) < \infty$).

Remarque 2: Si $a_k = (i,j)$, évaluer $f_k(X^p)$ exige qu'on calcule la longueur de la plus courte chaîne dans le réseau $(X^p - e_k)$. Pour ce calcul, l'algorithme de Dijkstra (1959) est le plus efficace.

Remarque 3: Dans l'exploration de l'arborescence associée, lorsqu'on effectue un retour en arrière, il arrive qu'on ait à recalculer des $f_j(X^p)$ ou

encore des $c_{ij}(X^P)$ déjà calculés. Pour éviter de refaire ces calculs déjà faits, on serait avisé de conserver en mémoire les valeurs des $f_j(X^P)$ déjà calculées; cela nécessite un nombre de mots de mémoire au plus égal à $m\ n$ (borne théorique), mais en fait beaucoup plus petit. Quant aux $c_{ij}(X^P)$, conserver tous ceux qui sont susceptibles d'être réutilisés exigerait trop d'espace-mémoire; l'astuce du programmeur a ici sa place: il pourra, par exemple, ne conserver que les $c_{ij}(X^P)$ qui changent effectivement d'une évaluation à la suivante.

Remarque 4: Finalement, nous voulons insister sur un aspect important des calculs. Dans l'algorithme tel que nous l'avons présenté, l'évaluation de $F(X^P)$ et des $f_j(X^P)$ dans le but de calculer la borne I^P se fait à chaque sommet de l'arborescence et elle exige un effort de calcul très important.

Pour réduire cet effort, on peut se contenter, comme le suggère Hoang (1971), de faire l'évaluation des $f_j(X^P)$ seulement à chaque fois qu'un certain nombre d'arêtes ont été exclues du réseau X^P, c'est-à-dire suivant une certaine fréquence (notation: FREQ). Au fur et à mesure de l'exploration de l'arborescence associée, on compte le nombre d'arêtes exclues (NEX) et on procède à l'évaluation des $f_j(X^P)$ seulement si

$$NEX \equiv 0 \pmod{FREQ}.$$

Hoang (1971) suggère aussi de ne pas poursuivre l'évaluation des $f_j(X^P)$ au-delà d'un certain niveau (niveau ≡ nombre de variables fixées) dans l'arborescence associée; nous appelons ce niveau NMAX.

L'évaluation des $f_j(X^P)$ exige la connaissance préalable de $F(X^P)$; on se contentera donc d'évaluer $F(X^P)$ lorsque $NEX \equiv 0 \pmod{FREQ}$ et lorsque X^P devient une solution réalisable (pour comparer $F(X^P)$ à $F(X^O)$).

En procédant ainsi, il arrive qu'on ne dispose pas des valeurs de $F(X^p)$ et des $f_j(X^p)$ pour calculer la borne I^p. On utilise alors à leur place les valeurs de F et des f_j les plus récemment calculées (ou récupérées de la mémoire si on procède comme indiqué à la remarque 3, ci-dessus).

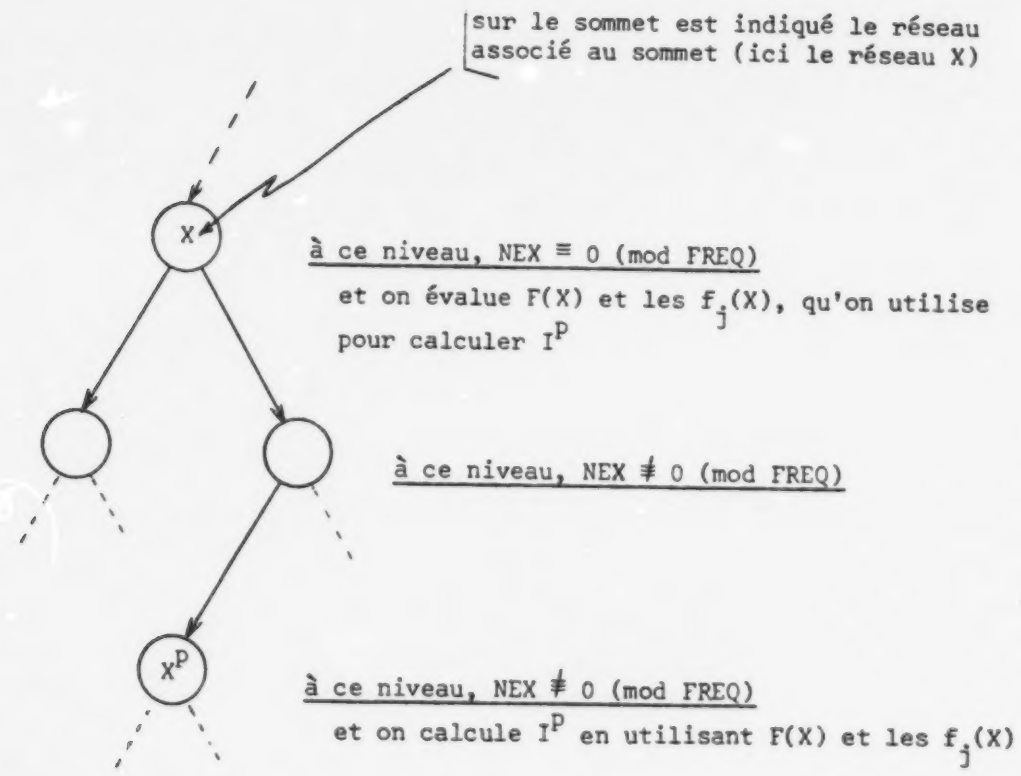


Figure 2 (exemple d'arborescence associée)

Si la plus récente évaluation de F et des f_j a été faite pour le réseau X, par exemple (voir figure 2), on a:

$$F(X) \leq F(X^P) \quad \text{et} \quad f_j(X) \leq f_j(X^P) \quad (j \in M) .$$

Et on calcule I^P en utilisant $F(X)$ à la place de $F(X^P)$ et $f_j(X)$ au lieu de $f_j(X^P)$.

La borne inférieure I^P devient ainsi plus faible et donc moins apte à effectuer une coupe, ce qui entraîne une augmentation du nombre de sommets de l'arborescence associée. Cependant, on espère réaliser une épargne en temps de calcul, étant donné qu'on évalue moins fréquemment F et les f_j. Le choix de FREQ et de NMAX est une question d'heuristique. Nous en reparlerons dans la troisième partie du présent chapitre, qui traite de la recherche expérimentale d'un algorithme plus efficace.

3.3 Une approche expérimentale à la recherche d'un algorithme plus efficace

3.3.1 Introduction

Le présent travail s'est trouvé amorcé par une idée très simple: dans l'algorithme de Hoang (1971), il paraît plausible que le remplacement de l'algorithme de Floyd-Murchland (défini par Floyd (1962)) par l'algorithme AS (défini et étudié au chapitre II)permette de diminuer le temps de calcul. Dans un réseau symétrique X dont on connaît la matrice C(X) , l'algorithme AS permet de repérer les plus courtes chaînes qui sont effectivement modifiées par la suppression d'une arête et de calculer leur nouvelle longueur plus efficacement que l'algorithme de Floyd-Murchland.

Cependant, dans l'algorithme de Hoang (1971), le calcul de la matrice C n'est pas effectué chaque fois qu'on supprime une arête, tandis que l'algorithme AS , lui, doit être mis en oeuvre après chaque suppression. Pour cette raison, on ne peut déterminer théoriquement avec certitude si le remplacement de l'algorithme de Floyd-Murchland par l'algorithme AS entraîne effectivement une diminution appréciable du temps de calcul. Nous avons donc eu recours à l'expérimentation pour trancher la question.

En cours de travail, nous avons vu d'autres possibilités, nous avons eu d'autres idées susceptibles de donner un algorithme encore plus efficace. Mais toujours la situation se présentait de la même façon: l'implantation d'une idée que nous estimions susceptible de diminuer le temps de calcul entraînait une modification de l'effort de calcul dont il était difficile, sinon impossible, d'évaluer l'impact sur le temps de

calcul; et il fallait recourir à l'expérimentation. Nous avons donc écrit (en FORTRAN FTN pour l'ordinateur CDC CYBER 74) un programme comportant diverses options de calcul, une option par idée à tester. Ce programme expérimental apparaît d'ailleurs dans l'appendice I. Comme données, nous disposions de huit réseaux, brièvement décrits dans le tableau 1 à la page suivante.

Les options de calcul que nous avons évaluées et comparées entre elles sont:

- option 1 : l'algorithme de Hoang (solution initiale par la méthode du graphe partiel connexe minimal)

- option 2 : la même que l'option 1, mais on a remplacé l'algorithme de Floyd-Murchland par l'algorithme AS

- option 3 : la même que l'option 2, sauf que le critère de choix de la variable séparatrice devient $(f_{j/\ell_j}, 1)$

- option 4 : la même que l'option 2, sauf qu'on prend $(Q_{j/\ell_j}, 1)$ comme critère de choix.

D'une option à la suivante, nous avons cherché à augmenter l'efficacité de l'algorithme initial (option 1), qui semble être lui-même une bonne méthode optimale (d'après Hoang (1971): "[...] the method [...] compares well with the existing optimal methods.").

3.3.2 Quelques remarques et notation supplémentaire

Il sera utile pour les fins de la discussion qui suit de donner ici quelques nouvelles définitions.

A un réseau donné, nous associons deux quantités, LMIN et LMAX, qui sont des caractéristiques du réseau, au même titre que m et n :

- LMIN = longueur totale des arêtes faisant partie du graphe partiel connexe minimal
- LMAX = longueur totale des arêtes dans $A = \sum_{j=1}^{m} \ell_j$.

Les huit réseaux que nous avons utilisés ont les caractéristiques suivantes:

nom du réseau	n	m	LMIN	LMAX	provenance
RH1	7	16	122	479	Hoang (1971), pp. 15-16 (longueur des arêtes multipliée par 2)
RH2	8	18	281	1041	
RH3	9	20	298	1122	
RB4	10	45	1654	20619	Boyce *et al.* (1973),p.531
R5	12	20	667	1944	réseaux que nous avons construits
R6	20	40	197	589	
R7	20	40	872	2722	
R8	29	54	1453	3654	

Tableau 1

Note: Les huit réseaux en question sont complètement décrits dans l'appendice II.

Pour une option de calcul donnée, nous considérerons dorénavant FREQ et NMAX comme des paramètres de cette option de calcul. De la même façon, nous considérerons B comme un paramètre (paramètre du problème). Cependant, plutôt que de travailler avec le paramètre B (budget), nous trouvons plus pratique d'utiliser un autre paramètre, IB , qui indique quel pourcentage du coût (longueur) maximum LMAX est accordé comme budget:

$$IB = 100 * B / LMAX .$$

En permettant, pour chaque option de calcul, la variation des paramètres, nous aurions généré un trop grand nombre de résultats numériques. Pour éviter cela, nous avons fixé a priori (mais en tenant compte d'une courte expérimentation préalable) la valeur de chacun des trois paramètres impliqués dans notre étude, en posant:

- FREQ = 3 , parce que l'expérimentation préalable avait indiqué que l'algorithme de Hoang est plus rapide pour de petites valeurs de FREQ (cf. remarque 4, page 64)
- NMAX > m , ce qui rend inopérant le paramètre NMAX , sur l'utilité duquel l'expérimentation préalable nous avait laissés fort sceptiques
- IB = 50 .

En ne donnant ainsi qu'une seule et même valeur à chaque paramètre, quels que soient l'option de calcul et le réseau considérés, nous facilitions grandement le travail de comparaison des résultats numériques obtenus avec les différentes options de calcul. Nous étions bien conscients cependant que cela comportait un danger: le choix de ces valeurs pour les para-

mètres pouvait favoriser une option de calcul aux dépens des autres. D'où la nécessité de faire (a posteriori) une analyse paramétrique des résultats, dont on trouvera un compte-rendu à la section 3.3.5.

Pour faciliter l'analyse des résultats numériques dans les pages qui suivent, nous les désignons par des identificateurs, dont la définition apparaît dans la table ci-dessous. On ne devrait pas avoir à consulter trop souvent cette table, car les identificateurs ont été choisis pour leur qualité mnémotechnique.

Table des identificateurs

- NSOM : désigne le nombre de sommets dans l'arborescence associée générée par l'option de calcul
- NAS : nombre de fois où l'algorithme AS est mis en oeuvre
- NFLOYD : nombre de fois où l'algorithme de Floyd-Murchland est mis en oeuvre
- NEVAL : nombre de fois où l'on procède à l'évaluation des f_j
- NBORNE : nombre de fois où l'on calcule la borne I^P
- UTIL% : mesure (en pourcentage) de l'utilité de la borne I^P, c'est-à-dire de sa capacité à effectuer une coupe:
  UTIL% = 100 * (nombre de fois où la borne coupe) / NBORNE
- MCOUPE : niveau moyen auquel la borne I^P effectue une coupe:
  $$\text{MCOUPE} = \frac{\text{somme des niveaux auxquels la borne coupe}}{\text{nombre de fois où la borne coupe}}$$
- NOPT : numéro du sommet où la solution optimale a été trouvée dans l'arborescence associée
- T : temps de calcul en secondes (à l'exclusion du temps pris pour calculer la solution initiale)
- T_{in} : temps de calcul de la solution initiale
- Z : valeur de la fonction économique à l'optimum
- Z_{in} : valeur de la fonction économique pour la solution initiale

Comme la donnée d'un réseau et d'une valeur pour le paramètre IB suffit à définir un problème de choix optimal d'un réseau, nous désignerons ci-après un tel problème par la formule (réseau, valeur de IB). Par exemple, (RH1,50).

3.3.3 Comparaison de l'option 1 et de l'option 2

Nous avons déjà expliqué, au tout début de la section 3.3, pourquoi nous espérions réduire le temps de calcul en remplaçant, dans l'algorithme de Hoang, l'algorithme de Floyd-Murchland (option 1) par l'algorithme AS (option 2). Nous donnons immédiatement les résultats numériques obtenus pour chacune de ces deux options.

problème → / ↓ identificateur		(RH1,50)	(RH2,50)	(RH3,50)	(RB4,50)	(R5,50)
NSOM	option 1	451	1853	4431	177	733
	option 2	209	938	1928	155	144
NFLOYD →	option 1	113	426	993	23	225
NAS →	option 2	80	365	775	52	62
NEVAL	option 1	19	111	146	18	2
	option 2	17	85	123	16	2
NBORNE	option 1	270	1131	2665	117	411
	option 2	129	573	1150	103	82
UTIL%	option 1	33.0	36.2	33.7	48.7	21.7
	option 2	38.0	36.3	32.3	49.5	24.4
MCOUPE	option 1	9.0	10.9	12.6	8.1	12.3
	option 2	8.2	9.7	10.6	7.9	8.8
NOPT	option 1	120	1418	831	15	6
	option 2	63	725	425	15	6
T(s.)	option 1	0.95	5.56	12.84	5.58	2.60
	option 2	0.47	3.11	6.11	4.96	0.53

Rappel: FREQ = 3 et NMAX > m .

Remarque: Les réseaux R6, R7 et R8 ne sont pas considérés ici à cause de leur temps de calcul prohibitif. Il en sera de même dans les tableaux 3,4 et 5.

Tableau 2

Dans le tableau 2, on aura remarqué que $NSOM(2)^* < NSOM(1)$. Mais pourquoi l'option 2 génère-t-elle ainsi une arborescence associée qui contient moins de sommets que l'option 1? Il n'y a qu'une explication possible. A chaque sommet de l'arborescence associée, correspond une solution X^P . Or l'option 1 évalue la matrice $C(X^P)$ et, à partir d'elle, $F(X^P)$ dans deux cas seulement: avant d'évaluer les f_j et lorsque X^P est une solution réalisable. Ainsi, la valeur exacte de $F(X^P)$ peut ne pas être disponible, dans l'option 1; on utilise alors à sa place la valeur la plus récemment calculée de la fonction économique F , disons F_a ($F_a \leq F(X^P)$) . Comme $F(X^P)$ constitue l'un des termes de la borne inférieure I^P , il en résulte pour l'option 1 une sous-évaluation de la borne, qui peut ne pas entraîner une coupe, alors que la valeur exacte de la borne aurait effectué une coupe. Si on compare UTIL%(1) et UTIL%(2) dans le tableau 2, on constate que la borne I^P coupe à peu près aussi fréquemment dans l'option 1 que dans l'option 2. Cependant, les coupes s'effectuent en moyenne plus bas dans l'arborescence associée pour l'option 1 que pour l'option 2 ($MCOUPE(1) > MCOUPE(2)$). Il en résulte que $NSOM(1) > NSOM(2)$, $NBORNE(1) > NBORNE(2)$ et $NEVAL(1) > NEVAL(2)$ pour tous les problèmes que nous avons traités.

On ne peut pas corriger ce défaut inhérent à l'option 1 (sous-évaluation de I^P) en calculant $C(X^P)$ à chaque sommet exploré, parce que l'algorithme de Floyd-Murchland est trop lent. Dans l'option 2, ce défaut est évité, car la matrice C est constamment tenue à jour par l'algorithme AS .

On aura aussi remarqué que l'utilité de la borne est de l'ordre de

---

* Note: La notation "NSOM(2)" signifie "NSOM pour l'option 2".

33%. C'est donc dire que calculer la borne I^P conduit une fois sur trois environ à effectuer une coupe, ce qui est considérable.

Expérimentalement, l'option 2 se révèle nettement plus efficace que l'option 1. En effet (à une exception près (NFLOYD(1) < NAS(2) pour RB4), qui est sans importance, puisque l'algorithme de Floyd-Murchland est plus lent que AS), tous les résultats numériques de l'option 2 sont plus intéressants que ceux de l'option 1, dans le tableau 2. Nous ne conservons donc que l'option 2, que nous allons maintenant comparer à l'option 3.

3.3.4 Comparaison de l'option 2 et de l'option 3

L'option 3 est obtenue de l'option 2 par simple modification du critère de choix de la variable séparatrice: dans l'option 3, on prend x_1 comme variable séparatrice plutôt que x_r comme dans l'option 2.

Cette modification se justifie facilement. On se rappellera qu'avant de résoudre le problème (2) (voir section 2.4), on réordonne les variables non encore fixées de façon à obtenir

$$\frac{f_1(X^P)}{\ell_1} \leq \frac{f_2(X^P)}{\ell_2} \leq \ldots \leq \frac{f_e(X^P)}{\ell_e} \quad ,$$

$$\text{alors que} \quad L^P = \{1,2,\ldots,e\} \quad .$$

Hoang suggère de choisir x_r comme variable séparatrice (r , le plus petit entier $\ni$ $0 < r \leq e$ et $\sum_{i=1}^{r} \ell_i > B - \sum_{\substack{j \in P \\ j>0}} \ell_j$) .

Nous voyons mal la motivation de son choix. En effet, si l'arête a_k s'écrit $a_k = (i,j)$, f_k représente l'allongement subi par la plus courte chaîne de i à j lorsqu'on supprime a_k . On peut donc considérer, que f_k/ℓ_k représente une mesure (relative et assez peu précise) de l'allongement par unité de longueur causé par la suppression de l'arête a_k . Dans ce cas, il serait plus approprié de choisir x_1 comme variable séparatrice, puisqu'à x_1 correspond la plus petite mesure d'allongement unitaire (f_1/ℓ_1) . Cette mesure, bien qu'imprécise, constitue un indice pouvant nous guider plus rapidement vers une solution optimale ou en tout cas vers des solutions réalisables plus proches de l'optimum, d'où une meilleure évaluation de la borne I^p et, par suite, une arborescence générée plus petite et un temps de calcul plus court. Voyons si les résultats numériques confirment nos dires.

problème → / ↓identificateur		(RH1,50)	(RH2,50)	(RH3,50)	(RB4,50)	(R5,50)
NSOM	option 2	209	938	1928	155	144
	option 3	134	781	188	62	234
NAS	option 2	80	365	775	52	62
	option 3	56	297	72	23	89
NEVAL	option 2	17	85	123	16	2
	option 3	8	69	17	6	19
NBORNE	option 2	129	573	1150	103	82
	option 3	78	483	115	39	145
UTIL%	option 2	38.0	36.3	32.3	49.5	24.4
	option 3	28.2	38.3	36.5	41.0	38.6
MCOUPE	option 2	8.2	9.7	10.6	7.9	8.8
	option 3	7.6	8.8	7.8	8.9	6.8
NOPT	option 2	63	725	425	15	6
	option 3	88	12	46	32	6
T(s.)	option 2	0.47	3.11	6.11	4.96	0.53
	option 3	0.30	2.30	0.74	1.75	1.08

Rappel: FREQ = 3 et NMAX > m .

Remarque: Nous reprenons ici les résultats du tableau 2 pour l'option 2 afin de faciliter le travail de comparaison.

Tableau 3

Pour les réseaux RH1, RH2, RH3 et RB4, les résultats numériques correspondent assez bien à ce que nous attendions de l'option 3:

- un optimum atteint plus rapidement (NOPT(3) << NOPT(2)), lorsque NOPT(2) est grand; et un optimum atteint après l'examen d'un nombre semblable de sommets (NOPT(3) ≅ NOPT(2)) , lorsque NOPT(2) est petit
- une arborescence associée plus petite (NSOM(3) < NSOM(2))
- et, par conséquent, un temps de calcul plus court (T(3) < T(2)) .

Pour le problème (R5,50) , l'option 3 ne fait pas mieux que l'option 2, mais il faut remarquer que l'arborescence associée générée par chacune des deux options est relativement petite (NSOM(2) = 144 , NSOM(3) = 234) et que NOPT(2) = NOPT(3) = 6 .

En suggérant de remplacer le critère de choix de la variable séparatrice $(f_j/\ell_j , r)$ par le critère $(f_j/\ell_j , 1)$, nous proposons une nouvelle heuristique, sans nous attendre à ce qu'elle soit meilleure dans tous les cas. Nous croyons cependant que la nouvelle heuristique proposée est, dans l'ensemble, meilleure que l'option 2. Et les résultats du tableau 3 nous confirment dans cette opinion. Il peut sembler que nous concluons à partir d'un petit nombre de résultats numériques, mais les résultats du tableau 3 ne sont pas les seuls que nous ayons recueillis: nous avons aussi obtenu des résultats pour les 20 problèmes définis par les réseaux RH1, RH2, RH3, RB4 et R5 et les budgets IB = 30, 40, 60 et 70. Ce sont toujours les mêmes tendances qui se dégagent. C'est pourquoi nous conservons l'option 3 comme la meilleure option jusqu'à maintenant.

3.3.5 Etude paramétrique de l'option 3

Pour atteindre une meilleure compréhension de l'option 3, il reste à étudier les paramètres FREQ et NMAX . Observons d'abord les effets du paramètre FREQ sur le comportement de l'algorithme.

	problème →	(RH1,50)	(RH2,50)	(RH3,50)	(R5,50)
	FREQ = 1	(NSOM=134,NEVAL=22) T =0.45	(693,170) 3.70	(176,42) 1.29	(228,54) 2.10
	FREQ = 2	(134,11) 0.32	(747,92) 2.74	(184,24) 0.87	(239,29) 1.37
	FREQ = 3	(134,8) 0.30	(781,69) 2.27	(188,17) 0.72	(234,19) 1.08
	FREQ = 4	(137,5) 0.28	(806,39) 1.99	(189,9) 0.57	(262,20) 1.10
NMAX ≥	FREQ = 5	(134,7) 0.27	(824,62) 2.26	(188,10) 0.57	(274,9) 0.90
	FREQ = 6	(134,5) 0.24	(851,49) 2.04	(188,12) 0.56	(268,1) 0.74
	FREQ = 7	(137,1) 0.22	(885,28) 1.76	(188,3) 0.45	
	FREQ = 8		(907,1) 1.56	(192,2) 0.45	
	FREQ = 9			(192,1) 0.41	

Remarques: 1) Chaque élément du tableau doit se lire: (NSOM, NEVAL) / T(s.)

2) Le réseau RB4 n'apparaît pas ici, parce que les résultats sont presque toujours les mêmes quelle que soit la valeur de FREQ .

Tableau 4

L'effet du paramètre FREQ , lorsque NMAX > m , ressort de façon très claire:

lorsque FREQ grandit, NSOM accuse une légère tendance à la hausse, cependant que NEVAL décroît franchement. Même si NSOM grandit un peu, les f_j sont évalués moins souvent et l'effet net apparaît: le temps de calcul T diminue quand FREQ grandit. Pour α suffisamment grand,T reste le même et NEVAL devient égal à 1 pour toute valeur de FREQ $\geq \alpha$.

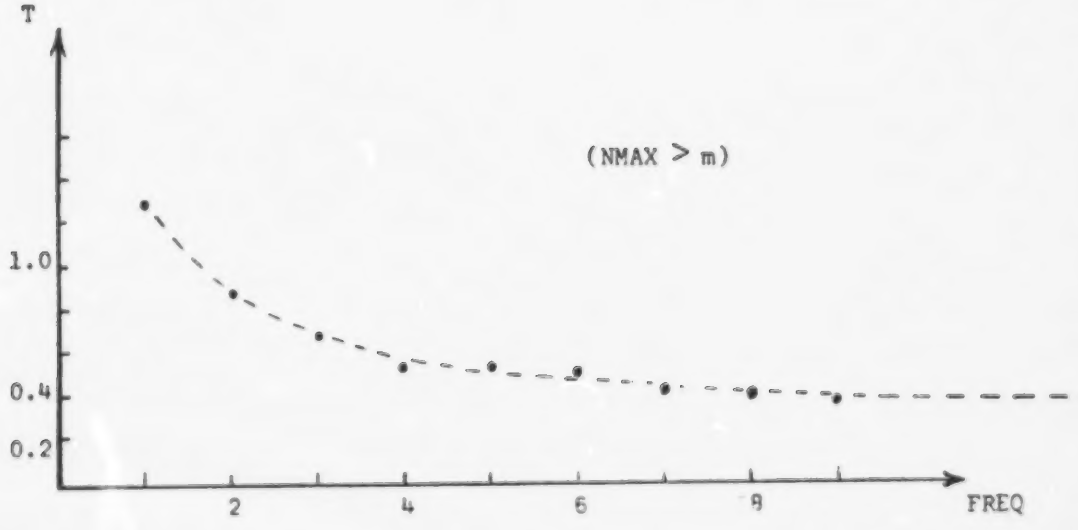


Figure 3

Graphique (T vs FREQ) pour l'option 3

(Note: Les valeurs numériques portées sur le graphique sont celles du tableau 5 pour le problème (RH3,50), mais l'allure de la courbe est la même pour n'importe quel problème.)

Les paramètres FREQ et NMAX jouent un rôle utile dans l'algorithme de Hoang (option 1). Mais l'option 3 diffère considérablement de l'option 1: les paramètres FREQ et NMAX n'y ont aucun rôle à jouer, puisque T atteint son minimum lorsqu'on évalue les f_j une seule fois avant de commencer à générer l'arborescence associée (NEVAL = 1) . Dès lors, nous supprimons les deux paramètres FREQ et NMAX de l'option 3.

Il y a un troisième paramètre, dont il est intéressant de connaître l'effet sur le temps de calcul: c'est le paramètre IB (paramètre du problème).

réseau →	RH1	RH2	RH3	RB4	R5
IB = 70	0.08	0.14	0.17	0.78	0.38
IB = 60	0.06	0.27	0.35	1.01	0.48
IB = 50	0.22	1.57	0.43	0.76	0.75
IB = 40	0.24	1.01	0.84	0.81	2.60
IB = 30	1.17	5.07	13.01	5.12	(LMIN > B)

T (s.) en fonction de IB

Tableau 5

Un fait ressort clairement: le temps de calcul augmente lorsque IB diminue. Boyce *et al.* (1973) ont d'ailleurs obtenu un résultat semblable pour leur algorithme.

Le type d'algorithme représenté par les diverses options de calcul que nous étudions se comporte bien lorsque le budget est peu restrictif, mais il lui faut beaucoup de temps pour résoudre un problème à petit budget. En effet, pour un petit budget, l'algorithme doit éliminer plusieurs arêtes du réseau initial R avant d'atteindre une solution X^p qui satisfasse la contrainte du budget; il doit donc descendre à un niveau plus

bas (i.e. un niveau de numéro plus grand) en générant l'arborescence associée. Or, dans une arborescence binaire à n sommets, si on ajoute 2 sommets comme successeurs à chaque sommet pendant, on se trouve à multiplier par 2 le nombre de sommets dans l'arborescence (plus précisément, cela donne 2n + 1 sommets). D'une façon analogue (pour l'arborescence associée), si on diminue le budget suffisamment, on descend 1 niveau plus bas en générant l'arborescence associée et on multiplie par 2 le nombre de sommets générés. Par suite, le temps de calcul (grosso modo proportionnel au nombre de sommets générés) a tendance à être proportionnel à

$$2^{-\,c\,*\,IB} \qquad \text{(c, une constante positive).}$$

Bien sûr, il ne s'agit là que d'une tendance, puisque la contrainte de connexité limite la génération des sommets et qu'il y a d'autres facteurs qui ont une influence sur le temps de calcul (par exemple, les sommets qu'on atteint après avoir posé une variable égale à 0 exigent plus de temps de calcul que ceux qu'on atteint après avoir fixé une variable égale à 1). Ce comportement exponentiel du temps de calcul est difficile à mettre en évidence numériquement pour de petits problèmes comme ceux du tableau 5, mais nous y reviendrons au chapitre suivant et nous l'y mettrons en évidence.

3.3.6 Comparaison de l'option 3 et de l'option 4

Nous avons vu à la section 3.2.1 que, pour un réseau donné X, $Q_k(X)$ constitue une meilleure mesure de l'utilité de l'arête a_k que $f_k(X)$. Or, dans l'option 3, les f_j jouent deux rôles importants:

- par l'intermédiaire du critère de choix $(f_j/\ell_j, 1)$, ils servent à déterminer le choix de la variable séparatrice et ils constituent ainsi un guide permettant d'atteindre plus rapidement la solution optimale
- et ils entrent dans le calcul de la borne I^P, qui contribue fortement à réduire le nombre de sommets dans l'arborescence associée.

Il serait donc souhaitable de faire jouer ces deux rôles par les Q_j, d'autant plus que l'algorithme AS peut être mis à contribution pour calculer rapidement les Q_j.

Les Q_j peuvent facilement remplacer les f_j dans la définition du critère de choix de la variable séparatrice: il suffit de prendre $(Q_j/\ell_j, 1)$ comme critère de choix. (Précisons bien ici que cette modification du critère de choix ne doit pas affecter la façon de calculer la borne I^P : pour le calcul de la borne, les variables continuent d'être ordonnées suivant l'ordre croissant des f_j/ℓ_j.)

Quant à l'utilisation des Q_j à la place des f_j dans le calcul de la borne I^P, elle s'avère impraticable. En effet, le théorème I n'est plus vrai si on y remplace f_j par Q_j.

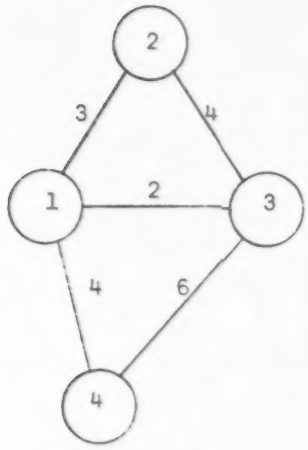
Enoncé du théorème I (rappel): $\forall X \in E^P$, on a:

$$F(X) \geq F(X^P) + \sum_{j \in L^P} \bar{x}_j \, f_j(X^P) .$$

Enoncé à infirmer: $\forall X \in E^P$, on a:

$$F(X) \geq F(X^P) + \sum_{j \in L^P} \bar{x}_j \, Q_j(X^P) .$$

Contrexemple:



Numérotons les arêtes: $a_1 = (1,2)$ $\ell_1 = 3$

$a_2 = (1,3)$ $\ell_2 = 2$

$a_3 = (1,4)$ $\ell_3 = 4$

$a_4 = (2,3)$ $\ell_4 = 4$

$a_5 = (3,4)$ $\ell_5 = 6$

Soient $X^p = (1,1,1,1,1)$

$X' = (0,1,1,1,1)$

$X'' = (0,1,0,1,1)$

$F(X^p) = 26$, $Q_1(X^p) = 6$ et $Q_3(X^p) = 7$

$F(X'') = 36 < F(X^p) + Q_1(X^p) + Q_3(X^p) = 39$

Fin du contrexemple.

Si l'on calcule la borne I^p en utilisant les Q_j à la place des f_j, il peut arriver que la borne I^p surestime la valeur de la fonction

économique de certaines solutions de E^P et on risque ainsi d'effectuer une coupe qui fera perdre la solution optimale. Nous utilisons cependant la borne I^P calculée à l'aide des Q_j dans une heuristique très efficace, qui calcule une solution réalisable toujours très proche de l'optimum et même presque toujours égale à l'optimum, et dont nous reparlerons dans le chapitre suivant. Cette nouvelle façon de calculer fournit une borne inférieure I^P beaucoup plus forte et donc beaucoup plus apte à couper l'arborescence associée, qui comporte alors beaucoup moins de sommets; d'où un temps de calcul réduit considérablement.

Nous devons donc nous contenter, dans notre tentative d'améliorer l'option 3, de faire jouer par les Q_j un seul des deux rôles joués par les f_j dans l'option 3. Nous proposons donc l'option 4, qui diffère de l'option 3 par le critère de choix de la variable séparatrice, qui devient $(Q_j/\ell_j, 1)$.

Puisque les Q_j constituent une meilleure mesure de l'utilité des arêtes que les f_j, nous nous attendons intuitivement à ce que les Q_j/ℓ_j se révèlent meilleurs guides que les f_{j/ℓ_j} dans le choix de la variable séparatrice et qu'ils nous conduisent plus rapidement vers l'optimum, entraînant ainsi une réduction du temps de calcul. Mais l'expérimentation a montré qu'il en va autrement, ainsi qu'on peut le constater en consultant le tableau 6. S'il est vrai que l'optimum a tendance à être atteint plus rapidement par l'option 4 (NOPT(4) du même ordre de grandeur ou bien plus petit que NOPT(3)), cela n'entraîne pas une diminution du temps de calcul pour l'option 4, étant donné que par la même

occasion, l'arborescence associée grandit (NSOM(4) > NSOM(3)) . Les arêtes ne sont pas éliminées dans le même ordre par le critère de choix de l'option 4 et celui de l'option 3. Or la valeur du terme $F(X^p)$ et la valeur du terme MIN $\sum_{j \in L^p} \bar{x}_j f_j(X^p)$ de la borne I^p sont toutes deux affectées par le choix des arêtes éliminées. Tout se passe comme si une plus grande concordance entre la mesure qui sert à définir le critère de choix de la variable séparatrice (i.e. f_j/ℓ_j pour l'option 3 et Q_j/ℓ_j pour l'option 4) et la mesure qui sert à calculer la borne I^p (i.e. f_j) contribuait à rendre la borne I^p plus apte à couper l'arborescence associée; cette concordance est plus grande pour l'option 3. L'explication que nous venons de donner est sûrement partielle, mais le phénomène échappe à l'analyse à cause de la complexité des interactions en jeu au cours des calculs.

L'option 3 demeure la meilleure option de calcul. C'est la meilleure méthode optimale que nous ayons mis au point.

	option 3				option 4			
↓ problème	NSOM	MCOUPE	NOPT	T (s.)	NSOM	MCOUPE	NOPT	T (s.)
(R6,65)	2498	14.9	1758	19.4	7856	15.9	1792	80.5
(R6,60)	3623	15.8	2647	28.2	10700	15.9	9	106.1
(R6,55)	4646	15.1	2619	35.9	22527	16.9	113	220.9
(R7,65)	530	9.9	9	3.8	1961	11.5	9	18.9
(R7,62)	3622	15.1	27	27.1	14856	16.1	19	146.0
(R7,60)	8363	17.4	107	63.3	43637	18.2	178	433.4
(R8,80)	1675	13.9	368	24.9	18902	16.8	10	353.9

Remarques: 1) Nous donnons ici des résultats obtenus pour les réseaux R6, R7 et R8. Pour les 5 autres réseaux (plus petits), les résultats ne diffèrent pas de façon significative de l'option 3 à l'option 4.

2) Nous rappelons que les paramètres FREQ et NMAX ont été éliminés de l'option 3. Après avoir vérifié que leur réintroduction n'améliore aucunement l'option 4, nous ne les avons pas non plus employés dans l'option 4.

Tableau 6

3.3.7 Influence de la solution initiale sur le temps de calcul

Nous n'avons jusqu'ici mis en oeuvre qu'un seul moyen pour calculer la solution initiale et c'est la méthode du graphe partiel connexe minimal décrite à la page 57.

Pour tenter d'améliorer les performances de l'option 3, il nous reste à lui faire calculer la solution initiale par une autre méthode, qui fournirait une meilleure solution initiale (meilleure en ce sens que Z_{in} serait plus proche de Z) et donc un meilleur point de départ pour les calculs. Une bonne heuristique pourrait très bien nous fournir une telle solution initiale. Nous considérerons au chapitre suivant quelques bonnes heuristiques. Nous utilisons dès maintenant la meilleure de ces heuristiques, que nous décrivons au chapitre suivant et que nous appelons la méthode MQO . Et nous modifions l'option 3 de façon qu'elle calcule la solution initiale par la méthode MQO .

Cette modification entraîne en général une légère diminution du temps de calcul T , diminution qui varie beaucoup d'un problème à l'autre (cf. tableau 7). La diminution de T est d'autant plus importante que la diminution de NOPT est plus grande. Quant au temps de calcul de la solution initiale (T_{in}) , à la suite de cette modification, il passe de 0.1 seconde environ à quelques secondes; à cause de cela, il peut parfois arriver qu'aucune économie ne soit réalisée sur le temps de calcul total $(T + T_{in})$.

Une autre information importante ressort des résultats numériques du tableau 7: l'option 3 se montre peu sensible à une variation de la solution initiale.

Option 3: solution initiale par MGPM — solution initiale par MQO

problème	NSOM	NOPT	T_{in}^{T} (≅ 0.1)	NSOM	NOPT	T_{in} + T = ...
(R6,65)	2498	1758	19.4	1437	1	1.2 + 11.0 = 12.2
(R6,60)	3623	2647	28.2	1453	1	0.7 + 10.9 = 11.6
(R6,55)	4646	2619	35.9	2622	1	0.7 + 19.5 = 20.2
(R6,53)	8566	3833	65.8	6229	1	0.8 + 46.5 = 47.3
(R6,50)	37117	9875	288.4	34877	1	1.8 +265.6 =267.4
(R7,65)	530	9	3.8	530	1	0.6 + 3.8 = 4.4
(R7,62)	3622	27	27.1	3622	1	1.2 + 26.9 = 28.1
(R7,60)	8363	107	63.3	8284	28	1.6 + 62.4 = 64.0
(R7,58)	26463	546	204.5	25971	1	1.9 +199.7 =201.6
(R8,80)	1675	368	24.9	1319	1	1.8 + 19.1 = 20.9
(R8,77)	2973	774	44.4	2261	1	3.0 + 32.9 = 35.9
(R8,75)	3458	1140	51.9	2357	1	1.6 + 34.3 = 35.9
(R8,73)	14745	1616	225.2	13171	1	2.7 +199.4 =202.1

Tableau 7

La meilleure option de calcul que nous ayons mis au point est donc l'option 3. Pour des problèmes de grande dimension comme ceux qu'on peut former avec les réseaux R6, R7 ou R8, on pourra faire calculer la solution initiale par la méthode MQO et réaliser ainsi une certaine économie de temps de calcul.

3.4 Conclusion

Le travail dont nous avons rendu compte dans le présent chapitre visait à mettre au point un algorithme optimal qui accroisse les capacités de calcul des méthodes optimales existantes pour le problème du choix optimal d'un réseau. Nous croyons y avoir réussi. En effet, comme l'indique un article de Sonia Stairs (1968) qui passe en revue la littérature sur le sujet, aucun chercheur ne rapporte de résultats numériques obtenus par une méthode optimale pour des réseaux de plus de 12 sommets. Depuis 1968, pour le cas sans congestion que nous avons traité, Scott (1969), Hoang (1971) et Boyce *et al.* (1973) sont les seuls chercheurs à avoir, à notre connaissance, présenté des algorithmes optimaux accompagnés de résultats numériques. Les résultats de Scott nous paraissent moins intéressants que ceux de Hoang et de Boyce *et al.* Et notre meilleur algorithme, l'option 3, peut avantageusement soutenir la comparaison avec les algorithmes de Hoang et de Boyce *et al.*

L'option 3 travaille plus rapidement que l'algorithme de Hoang (option 1); cela a été clairement établi dans les pages qui précèdent. Elle permet de résoudre des problèmes qui restent tout-à-fait inaccessibles à l'algorithme de Hoang: par exemple, pour le problème (R6,65), l'algorithme de Hoang n'a pas terminé ses calculs après 500 secondes, alors que l'option 3 n'exige que 12.2 secondes (tableau 7).

Dans le cas de l'algorithme de Boyce, nous ne pouvons actuellement faire des comparaisons que pour le réseau RB4 (le seul réseau complètement décrit dans la publication de Boyce *et al.*). De plus, ces comparaisons manquent un peu de précision, étant donné que Boyce *et al.* ont travaillé

sur un ordinateur IBM 360/75, alors que nous disposions d'un CDC CYBER 74. Nous estimons cependant que les temps de calcul sur l'un et l'autre ordinateur sont du même ordre de grandeur* et nous donnons les résultats suivants à titre indicatif:

pour le problème (RB4,70), T (Boyce) ≅ 2 s. et T (option 3) = 0.8 s.

pour le problème (RB4,50), T (Boyce) ≅ 23 s. et T (option 3) = 0.8 s.

pour le problème (RB4,30), T (Boyce) > 1,000 s. et T (option 3) = 5.1 s.

On aura remarqué que le temps de calcul croît beaucoup plus rapidement pour l'algorithme de Boyce que pour l'option 3, ce qui est indépendant de l'ordinateur utilisé.

Nous estimons avoir obtenu les meilleurs résultats numériques publiés jusqu'à maintenant.

---

* Note: Pour un IBM 360/75, la durée d'un cycle mémoire est de 0.75 microseconde et l'addition binaire exige 1.4 microseconde. Pour un CYBER 74, la durée d'un cycle mémoire est de 1.0 microseconde et l'addition binaire prend 0.4 microseconde. Nous tirons ces renseignements du COMPUTER REVIEW 1974 de GML Corporation, Lexington, Massachusetts.

CHAPITRE IV

QUELQUES HEURISTIQUES
POUR LE PROBLEME DU CHOIX OPTIMAL D'UN RESEAU

Sous heuristique, le dictionnaire dirait:

n.f. Art de faire des découvertes, particulièrement des découvertes de documents.

En recherche opérationnelle, le mot a pris le sens de recherche systématique et organisée, parmi un vaste ensemble de solutions, d'une solution réalisable qui soit, le plus possible, voisine de la solution optimale. Le principal avantage d'une heuristique réside dans son temps de calcul très court (beaucoup plus court que celui d'une méthode optimale). En contrepartie, une heuristique ne garantit pas l'optimalité de la solution qu'elle fournit. Cependant, une bonne heuristique donne une solution réalisable très proche de la solution optimale et même souvent égale à celle-ci.

Nous avons jusqu'ici rencontré deux heuristiques, les deux méthodes d'obtention d'une solution initiale décrites à la page 57:

- la méthode du graphe partiel connexe minimal (ci-après, heuristique MGPM)
- et la méthode des Q_j (ci-après, heuristique MQJ).

Un peu plus loin (sections 4.3 et 4.4), nous proposerons deux nouvelles heuristiques:

- l'heuristique MQJM (méthode des Q_j modifiée)
- et l'heuristique MQO ("méthode quasi optimale").

Dans le présent chapitre, nous voulons procéder à une brève évaluation de chacune de ces quatre heuristiques. Nous considérons dorénavant ces heuristiques comme fournissant une solution terminale ayant une valeur en soi, alors qu'auparavant nous les regardions comme fournissant une solution initiale servant d'amorce à un algorithme optimal. Nos critères d'évaluation sont le temps de calcul, que nous désignons par T (rompant ainsi avec l'usage que nous faisions de T au chapitre précédent), et l'écart relatif entre la solution fournie par l'heuristique et la solution optimale. Cet écart relatif est mesuré par

$$Er = \frac{Z_{heur} - Z}{Z_{heur}} ,$$

où Z désigne toujours la valeur de la fonction économique à l'optimum

et Z_{heur} désigne la valeur de la fonction économique pour la solution fournie par l'heuristique.

Notre évaluation est fondée sur de nombreux résultats numériques.

4.1 L'heuristique MGPM

Le seul avantage de la méthode MGPM réside dans son temps de calcul extrêmement court, pour ne pas dire négligeable (voir les valeurs de T dans le tableau 1, ci-après). De plus ce temps de calcul est indépendant du budget pour un problème donné.

Par contre, la solution fournie par cette heuristique s'avère fort mauvaise, l'écart relatif Er pouvant facilement atteindre 3 à 6% (cf. tableau 1).

4.2 L'heuristique MQJ

Nous avons déjà présenté l'heuristique MQJ ou méthode des Q_j , à la page 57. Cette heuristique a été proposée indépendamment par Scott (1969, page 208) et par Hoang (1971, page 13). Hoang suggère de l'utiliser pour calculer une bonne solution initiale pour son algorithme de choix optimal d'un réseau (option 1); il ne procède à aucune évaluation de l'heuristique. Scott évalue son heuristique en résolvant 26 problèmes de choix optimal d'un réseau; les réseaux considérés comportent au plus 10 sommets. Pour chacun des 26 problèmes, Scott rapporte un temps de calcul inférieur à une minute sur un ordinateur IBM 360/65; 24 des 26 solutions calculées par l'heuristique étaient optimales, les deux autres s'écartant de l'optimum de 0.03% et 2.3%.

Nous avons mené notre propre expérimentation. L'heuristique MQJ s'avère très rapide: pour tous les problèmes solutionnés (cf. tableau 1), le temps de calcul T n'est qu'affaire de quelques secondes. Par contre, sous le rapport de la proximité à l'optimum de la solution fournie, nous croyons que l'heuristique laisse à désirer. En effet, sur 18 solutions calculées par l'heuristique, 8 seulement sont optimales, et trop souvent l'écart relatif Er se situe entre 0.2% et 0.9% (7 cas sur les 18). Nous avons tenté de remédier à cette faiblesse, ce qui nous a amené à modifier l'heuristique MQJ, pour obtenir l'heuristique MQJM.

		heuristique MGPM			heuristique MQJ		
problème	Z	Z_{heur}	T	Er	Z_{heur}	T	Er
(RH1,50)	761	764	0.03	0.4%	761	0.10	0
(RH2,50)	2246	2354	0.03	4.6%	2246	0.15	0
(RH3,50)	2707	2775	0.03	2.5%	2732	0.22	0.9%
(RB4,50)	20696	20873	0.03	0.8%	20700	0.81	0.02%
(R5,50)	11945	12335	0.03	3.2%	11945	0.38	0
(R6,65)	5460	5510	0.08	0.9%	5474	3.03	0.2%
(R6,60)	5478	5659	↓	3.2%	5491	3.33	0.2%
(R6,55)	5503	5707		3.5%	5503	4.27	0
(R6,53)	5521	5707		3.3%	5535	4.22	0.3%
(R6,50)	5563	5806		4.2%	5585	4.52	0.4%
(R7,65)	30705	32254	0.07	4.8%	30871	3.86	0.6%
(R7,62)	30832	32295	↓	4.5%	30832	4.03	0
(R7,60)	30924	32295		4.2%	30932	4.11	0.04%
(R7,58)	31070	32320		3.9%	31070	3.98	0
(R8,80)	99611	103334	0.14	3.6%	99661	10.27	0.05%
(R8,77)	99698	103421	↓	3.6%	99738	10.24	0.4%
(R8,75)	99752	106301		6.2%	99752	11.32	0
(R8,73)	99914	106301		6.0%	99914	12.21	0

Tableau 1

4.3 L'heuristique MQJM

On peut apporter une modification très simple à l'heuristique MQJ et l'améliorer ainsi d'une façon considérable. La modification consiste à utiliser les Q_j/ℓ_j plutôt que simplement les Q_j. L'étape (ii)b) de la méthode MQJ (cf. page 57) devient alors:

(ii)b)... trouver k $\ni$ $\dfrac{Q_k(X)}{\ell_k} = \min_{\{j \in M \mid x_j=1\}} \dfrac{Q_j(X)}{\ell_j}$ et poser $x_k = 0$.

Cette modification permet de tenir compte du coût (ℓ_j) d'une arête (a_j) en même temps que de sa tendance à faire augmenter la longueur du réseau si on l'enlève (Q_j) ; les Q_j/ℓ_j mesurent l'utilité par unité de longueur et permettent une meilleure comparaison entre les arêtes susceptibles d'être éliminées.

Nous avons appelé MQJM l'heuristique MQJ ainsi modifiée et nous l'avons utilisée pour résoudre les mêmes 18 problèmes qu'au tableau 1. Les résultats apparaissent dans le tableau 2 et nous les jugeons impressionnants:

- légère réduction du temps de calcul par rapport à l'heuristique MQJ
- la solution calculée par MQJM est optimale dans tous les cas sauf deux (pour (R6,65), Er = 0.04% et pour (R8,80), Er = 0.003%).

La nouvelle heuristique MQJM présente donc les deux principales qualités qu'on attend d'une bonne heuristique: elle exige peu de temps

de calcul et elle produit une solution égale à l'optimum, sinon extrêmement voisine de l'optimum.

L'heuristique MQO

Nous proposons ici une heuristique complètement originale. On se souviendra qu'à la section 3.3.6, nous avions été frustrés dans notre tentative de produire une option de calcul calquée sur l'option 3, mais où les f_j seraient remplacés par les Q_j. En effet, si l'on calcule la borne I^P en utilisant les Q_j à la place des f_j, il peut arriver qu'on obtienne une borne inférieure surestimée, en ce sens que

$$I^P > \min \{F(X) \mid X \in E^P \text{ et } X \text{ réalisable}\} .$$

En pareil cas, l'utilisation de la borne I^P peut entraîner la perte de la solution optimale. Cependant, à la lumière de toute l'expérimentation que nous avons menée, nous estimons que la probabilité que l'optimum soit ainsi éliminé demeure très faible. C'est pourquoi nous proposons comme heuristique la méthode MQO, qui n'est autre que l'option 3 où les Q_j remplacent complètement les f_j.

Cette nouvelle heuristique s'avère encore plus efficace que l'heuristique MQJM pour traiter les 18 problèmes-tests: comme on pourra le constater en consultant le tableau 2, MQO est encore plus rapide que MQJM pour produire une solution réalisable presque toujours égale à l'optimum (une seule exception: pour (R7,60), Er = 0.03%).

		heuristique MQJM			heuristique MQO		
problème	Z	Z_{heur}	T	Er	Z_{heur}	T	Er
(RH1,50)	761	761	0.10	0	761	0.20	0
(RH2,50)	2246	2246	0.15	0	2246	0.63	0
(RH3,50)	2707	2707	0.17	0	2707	0.32	0
(RB4,50)	20696	20696	0.79	0	20696	0.43	0
(R5,50)	11945	11945	0.36	0	11945	0.20	0
(R6,65)	5460	5462	2.45	0.04%	5460	1.09	0
(R6,60)	5478	5478	2.74	0	5478	0.62	0
(R6,55)	5503	5503	3.32	0	5503	0.62	0
(R6,53)	5521	5521	3.56	0	5521	0.75	0
(R6,50)	5563	5563	4.17	0	5563	1.75	0
(R7,65)	30705	30705	2.62	0	30705	0.61	0
(R7,62)	30832	30832	3.18	0	30832	1.17	0
(R7,60)	30924	30924	3.48	0	30932	1.53	0.03%
(R7,58)	31070	31070	4.00	0	31070	1.86	0
(R8,80)	99611	99614	8.22	0.003%	99611	1.55	0
(R8,77)	99698	99698	9.20	0	99698	2.93	0
(R8,75)	99752	99752	10.17	0	99752	1.50	0
(R8,73)	99914	99914	12.11	0	99914	2.56	0

Tableau 2

4.5 Les heuristiques MQJM et MQO vues comme alternatives aux méthodes optimales

La nature hautement combinatoire du problème du choix optimal d'un réseau constitue un obstacle à l'obtention d'un algorithme optimal efficace dans la résolution des problèmes de grande taille (plusieurs dizaines de sommets): pour de tels problèmes, le temps de calcul devient facilement prohibitif. A la lumière de toute l'expérimentation que nous avons menée, nous croyons que les facteurs qui contribuent à allonger le temps de calcul s'établissent comme suit:

- 1) Le paramètre IB influence grandement le temps de calcul T . En effet, pour le type d'algorithme optimal que nous avons étudié au chapitre précédent, c'est-à-dire pour les options 1 à 4, le temps de calcul T grandit exponentiellement à mesure que IB décroît:

  T est approximativement proportionnel à $2^{-c * IB}$ (c, une constante).

  Ce comportement exponentiel est une caractéristique des P.S.E. que nous avons étudiées, et nous avons expliqué son origine au chapitre précédent (pages 82 et 83). Nous voulons maintenant mettre en évidence ce comportement exponentiel à l'aide de résultats numériques. L'heuristique MQO possède la même structure algorithmique que les options 1 à 4; c'est une P.S.E. au même titre que chacune des options 1 à 4. Le temps de calcul de MQO devrait donc lui aussi suivre une tendance exponentielle. Pour mettre tout cela en évidence, nous avons traité par MQO les réseaux R6, R7 et R8

en faisant varier le paramètre IB de 90 à 40. Il ressort de nos résultats numériques (cf. tableau 3, ci-après) que, pour un réseau donné, une fonction du type

$f(IB) = a\, b^{(100 - IB)/10}$, où a et b sont des constantes positives*, constitue un modèle qui prédit assez bien la valeur de T.

- 2) Le nombre n est aussi un facteur qui a une influence directe, quoique moins importante que celle de IB , sur le temps de calcul. En effet, à chaque sommet de l'arborescence associée, on calcule la matrice C_{nxn} . Heureusement, dans notre meilleur algorithme optimal (l'option 3), ce calcul est effectué par l'algorithme AS, qui exige beaucoup moins d'opérations élémentaires que l'algorithme de Floyd-Murchland (cf. chapitre II). Même si n était de l'ordre de plusieurs dizaines, le temps de calcul resterait abordable à condition que le nombre de sommets dans l'arborescence associée (NSOM) reste petit, ce qui se produit lorsque le budget n'est pas trop restrictif (i.e. lorsque IB est grand).

- 3) Le nombre d'arêtes, m , a aussi une certaine importance, ainsi que l'illustrera l'exemple suivant. Soient deux réseaux, R et R' , qui comportent le même nombre de sommets, mais supposons que R comporte 2 ou 3 fois plus d'arêtes que R' . Supposons de plus que toutes les arêtes (des 2 réseaux) ont la même longueur. Enfin supposons donnée une valeur pour le paramètre IB (la même pour

*Note: f(IB) est proportionnelle à $2^{-c * IB}$.

les 2 réseaux). Alors il est clair qu'en cherchant à résoudre le problème (R,IB) , l'algorithme devra, avant d'obtenir une solution qui satisfasse la contrainte du budget, éliminer plus d'arêtes qu'en cherchant à résoudre (R',IB). D'où une arborescence associée plus grande et un temps de calcul plus long.

- 4) Enfin, la topologie du réseau initial R fait que les plus courtes chaînes sont plus ou moins difficiles à calculer.

La recherche d'une solution optimale exige un temps de calcul qui peut, dans certains cas, devenir prohibitif, même pour le meilleur algorithme optimal que nous ayons mis au point, l'option 3. Des heuristiques comme MQJM et MQO constituent alors une alternative intéressante pour traiter les problèmes dont le degré de difficulté dépasse les capacités des méthodes optimales. En effet, ces deux heuristiques sont peu coûteuses en temps de calcul et elles fournissent une solution réalisable étonnamment proche de l'optimum. Il reste à voir quels sont les avantages respectifs de MQJM et de MQO.

D'après nos résultats numériques, MQJM et MQO paraissent aussi aptes l'une que l'autre à produire une solution pour laquelle Er soit nul ou très petit. De plus, l'heuristique MQO s'est révélée la plus rapide pour traiter les 18 problèmes du tableau 2 du présent chapitre. Cependant, le temps de calcul pour MQO croît exponentiellement à mesure que IB décroît, comme nous l'avons mentionné ci-dessus. Et ce n'est pas le cas pour MQJM. En effet, si l'on revoit la définition de l'heuristique MQJM (page 57), on constate que son temps de calcul est directement

proportionnel au temps d'évaluation des Q_j multiplié par le nombre d'arêtes éliminées du réseau initial R (nous négligeons ici l'étape (iv) qui ne constitue qu'un réajustement rapide de la solution trouvée). Grosso modo, le temps de calcul dépend donc linéairement du nombre d'arêtes éliminées, ce qui présente un avantage très net, par rapport à l'heuristique MQO, pour de petites valeurs de IB . Les résultats numériques du tableau 3, qui suit, illustrent les variations du temps de calcul T , suivant IB , pour MQJM et pour MQO.

Pour augmenter ses chances d'obtenir une solution optimale pour un problème donné, on pourra le résoudre à l'aide des deux heuristiques MQO et MQJM et ne conserver que la meilleure des deux solutions obtenues.

	MQJM				MQO			
problème	Z_{heur}	T	NEL	Δ	Z_{heur}	T	$f(IB)=ab^{(100-IB)/10}$	
(R6,90)	5398	0.99	3	0	5398	0.42	0.42	
(R6,80)	5408	1.56	4	+1	5407	0.61	0.62	
(R6,70)	5436	2.17	7	0	5436	0.83	0.89	a = 0.31
(R6,60)	5478	2.73	9	0	5478	0.59	1.24	b = 2
(R6,50)	5563	4.19	12	0	5563	1.76	1.75	
(R6,40)	5705	5.27	18	0	5705	2.45	2.48	
(R7,90)	30465	0.93	3	0	30465	0.37	0.25	
(R7,80)	30489	1.49	5	0	30489	0.39	0.50	
(R7,70)	30664	2.65	8	+31	30633	0.99	1.00	$a = \frac{1}{4}$
(R7,60)	30924	3.50	12	-8	30932	1.55	2.00	b = 2
(R7,50)	31724	4.35	14	-14	31738	5.20	4.00	
(R7,40)	32928	5.10	17	0	32928	8.71	8.00	
(R8,90)	99420	5.18	3	+8	99412	1.39	2	
(R8,80)	99614	8.21	7	+3	99611	1.58	4	
(R8,70)	100431	14.21	14	+70	100361	9.25	8	a = 1
(R8,60)	101959	17.30	16	-350	102309	16.47	16	b = 2
(R8,50)	104682	23.00	22	0	104682	17.10	32	
(R8,40)	132288	26.86	26			>300.	64	

Remarques: Pour R6, R7 et R8, $B < LMIN$ lorsque $IB \leq 30$.

Δ = (Z_{heur} pour MQJM) - (Z_{heur} pour MQO)

NEL = nombre d'arêtes éliminées.

Tableau 3

CHAPITRE V

EXTENSIONS

Le modèle mathématique que nous avons utilisé jusqu'ici pour le problème du choix optimal d'un réseau représente la réalité d'une façon plutôt simplifiée. Il serait souhaitable de le modifier pour qu'il s'applique dans un plus grand nombre de situations concrètes.

5.1 Généralisations

Dans le présent chapitre, nous proposons les généralisations suivantes:

- 1) réseau préexistant

Dorénavant, nous considérons que le problème du choix optimal d'un réseau consiste à choisir non plus un réseau optimal qu'il faudra ensuite construire de toutes pièces, mais plutôt quelle amélioration optimale on peut apporter à un réseau qui existe déjà en construisant de nouvelles routes (arêtes).

Formellement,

$A = \{a_1, a_2, \ldots, a_m\}$ = ensemble des arêtes

$A^* = \{a_1, a_2, \ldots, a_\mu\}$ = ensemble des arêtes distinguées

$A^* \subseteq A$, $0 \leq \mu \leq m$.

Les arêtes distinguées sont considérées a priori comme inclues dans tout réseau qu'on évaluera. Elles représentent les tronçons

de route déjà construits et constituent le réseau préexistant. Les autres arêtes représentent des projets de constructions, parmi lesquels il faut choisir de façon optimale pour améliorer le réseau préexistant tout en respectant le budget. Le réseau préexistant n'est pas nécessairement connexe.

- 2) coûts de construction

Le coût de construction d'une arête a_j peut avoir une valeur quelconque; il n'a plus nécessairement une valeur égale à sa longueur, ℓ_j.

Formellement,

c_j = coût de construction de l'arête a_j , $j \in M$

$c_j > , = \text{ ou } < \ell_j$

La motivation de cette généralisation est évidente: le coût de construction d'une route varie suivant son emplacement, pour une longueur donnée.

- 3) demandes O-D

La demande de transport n'est pas nécessairement la même entre toutes les paires O-D. Jusqu'ici, nous avons considéré comme également importantes toutes les paires O-D; ce n'est plus le cas. Formellement,

δ_{ij} = demande de transport entre le sommet i et le sommet j

$\Delta = (\delta_{ij})_{nxn}$ = matrice des demandes O-D

Pour retrouver le cas particulier que nous étudions auparavant, il suffit de considérer que

$$\mu = 0 \qquad (A^* = \emptyset)$$

$$c_j = \ell_j \quad , \quad \forall j \in M$$

$$\text{et} \quad \delta_{ij} = 1 \quad , \quad \forall i \in N \quad \text{et} \quad \forall j \in N \, .$$

Nous allons maintenant reformuler le problème du choix optimal d'un réseau, après avoir établi quelques définitions. Utilisant la notation introduite au début du chapitre III, nous définissons:

Pour tout réseau X ,

$$F^{\delta}(X) = \sum_{\substack{i,j \\ i<j}} \delta_{ij} \, c_{ij}(X) \, .$$

Si l'arête $a_k = (i,j)$ fait partie du réseau X ,

$$f_k^{\delta}(X) = \delta_{ij} \, f_k(X) = \delta_{ij} \, [\, c_{ij}(X - e_k) - c_{ij}(X) \,]$$

$$\text{et} \quad Q_k^{\delta}(X) = F^{\delta}(X - e_k) - F^{\delta}(X) \, .$$

Les définitions de F^{δ}, f_k^{δ} et Q_k^{δ} généralisent de façon naturelle les définitions de F , f_k et Q_k .

Le problème à résoudre s'énonce maintenant comme suit:

Etant donné un réseau R dans lequel on a distingué

un sous-ensemble $A^* = \{a_1, a_2, \ldots, a_\mu\}$ d'arêtes

parmi l'ensemble $A = \{a_1, a_2, \ldots, a_m\}$ de toutes les arêtes,

étant donné aussi un budget B ,

trouver R_p , un réseau partiel de R , tel que:

- R_p est connexe (contrainte 1)
- $\sum_{j=\mu+1}^{m} c_j x_j^p \leq B$ (contrainte 2)
- $x_j = 1$, $j = 1,2,\ldots,\mu$ (contrainte 3)
- $F^\delta(X^p)$ est minimum.

Parmi les algorithmes étudiées aux chapitres III et IV, retenons-en trois (les autres sont peu intéressants):

- notre meilleur algorithme optimal, l'option 3
- les deux meilleures heuristiques, MQJM et MQO.

Nous allons maintenant voir comment on peut modifier ces trois algorithmes de façon à ce qu'ils résolvent la nouvelle version du problème du choix optimal d'un réseau, et cela sans augmenter de beaucoup l'effort de calcul.

5.2 Modifications à apporter à l'heuristique MQJM

Essentiellement, dans sa première version, l'algorithme MQJM travaille avec les Q_j , enlevant l'arête a_k dont la suppression engendre la plus petite augmentation (Q_k) de la fonction économique, puis en itérant.

Q_k^δ mesure l'augmentation de la nouvelle fonction économique F^δ causée par la suppression de l'arête a_k . La nouvelle version de l'algorithme MQJM travaillera donc avec les Q_j^δ et elle s'énonce comme suit:

(i) Poser $X = (1,1,\ldots,1)$.

(ii) Soit $J = \{j \in M \mid j > \mu \quad \text{et} \quad x_j = 1\}$.

$\forall j \in J$, évaluer $Q_j^\delta(X)$. Puis

a) si $Q_j^\delta(X) = \infty \ \forall j \in J$, recommencer les calculs en employant la méthode du graphe partiel connexe minimal (à laquelle on aura fait subir les modifications évidentes qui s'imposent);

b) sinon, trouver $k \ni \dfrac{Q_k^\delta(X)}{c_k} = \min\limits_{j \in J} \dfrac{Q_j^\delta(X)}{c_j}$

et poser $x_k = 0$. Aller en (iii).

(iii) Si $\sum_{j=\mu+1}^{m} c_j x_j > B$, aller en (ii). Sinon aller en (iv).

(iv) Si le budget le permet, tenter de redonner à une composante nulle de X la valeur 1 de façon à minimiser $F^{\delta}(X)$. Répéter jusqu'à ce qu'il ne soit plus possible de modifier aucune composante de X. X est alors la solution cherchée.

5.3 Modifications à apporter à l'option 3

Certaines modifications s'imposent comme évidentes: dans la nouvelle version de l'option 3, il faudra utiliser les fonctions F^{δ} et f_k^{δ} à la place des fonctions F et f_k . De même, il faudra employer $\sum_{j=\mu+1}^{m} c_j x_j$ au lieu de $\sum_{j=1}^{m} \ell_j x_j$ pour évaluer le coût de construction d'un réseau $X = (x_1, x_2, \ldots, x_m)$.

Il faudra aussi modifier la première étape de l'option 3, c'est-à-dire l'initialisation. Tout d'abord, la solution initiale X^o devra être conforme à la nouvelle formulation ($x_j^o = 1$, $j \leq \mu$). On pourra utiliser la nouvelle version de l'heuristique MQJM pour calculer X^o . Les autres étapes de l'initialisation devront être modifiées comme suit:

- On initialisera la pile P de façon que sa représentation soit

$$P = (1, 2, \ldots, \mu, -, -, \ldots, -)$$

$$\text{ptr} = \mu + 1$$

et on donnera initialement au pointeur de la pile P la valeur $\mu + 1$. Cela permettra de tenir compte du fait que les arêtes $a_1, a_2, \ldots, a_\mu$ sont inclues a priori dans la solution optimale. Le pointeur ptr ne sera autorisé à parcourir que les valeurs $\mu + 1, \mu + 2, \ldots, m$ et l'algorithme s'arrêtera lorsque le pointeur ptr reviendra prendre la valeur $\mu + 1$ (et non plus la valeur 1 comme auparavant).

- C'est donc dire que, lors de l'initialisation,

  $L^p := \{\mu + 1, \mu + 2, \ldots, m\}$.

- $X^p := (1,1,\ldots,1)$.

Ces modifications nous permettent de préserver la structure algorithmique de l'option 3 : dans la nouvelle version de l'option 3, le déroulement des calculs restera le même que dans la version d'origine. Cela est rendu possible par le fait que la borne inférieure I^p reste valide si on la calcule avec les fonctions F^δ et f_k^δ à la place des fonctions F et f_k, comme nous allons le démontrer maintenant.

Théorème II : $\forall X \in E^p$, $X = (x_1, x_2, \ldots, x_m)$, on a :

$$F^\delta(X) \geq F^\delta(X^p) + \sum_{j \in L^p} \overline{x}_j \, f_j^\delta(X^p) .$$

Preuve: Le théorème II est la généralisation du théorème I du chapitre III et la preuve du théorème II procède de la même façon que celle du théorème I .

Soit $X \in E^p$.
Excluons le cas où $F^\delta(X) = \infty$, car alors le théorème se vérifie trivialement. Supposons donc X connexe.

$X \in E^p \iff X$ est obtenu de X^p par suppression d'un certain nombre d'arêtes

Appelons ce nombre s ($s \leq m - \mu$) . Sans perte de généralité, supposons que les s arêtes supprimées soient $a_{\mu+1}, a_{\mu+2}, \ldots, a_{\mu+s}$.

$$(*)\quad \begin{cases} x_j = 1\ ,\ x_j^p = 1\ , & \text{pour } j \leq \mu \\ x_j = 0\ ,\ x_j^p = 1\ , & \mu + 1 \leq j \leq \mu + s \\ x_j = x_j^p\ , & \text{pour } j > \mu + s \end{cases}$$

Note: $L^p \supseteq \{\mu + 1, \mu + 2, \ldots, \mu + s\}$

et $\forall j \in L^p \ni j > \mu + s$, on a: $x_j = x_j^p = 1$.

Supposons que le réseau Y diffère du réseau X par le seul fait qu'il contient une arête de plus que X , disons l'arête $a_{\mu+s}$.

$$(**)\quad X = Y - e_{\mu+s} \quad \text{i.e.} \quad \begin{cases} y_{\mu+s} = 1\ , & \text{mais } x_{\mu+s} = 0\ , \\ y_j = x_j\ , & \forall\ j \in M - \{\mu + s\}\ . \end{cases}$$

$$(***)\quad F^{\delta}(X) \geq F^{\delta}(Y) + f^{\delta}_{\mu+s}(Y)\ .$$

En effet, si $a_{\mu+s} = (u,v)$, $u < v$,

alors $\delta_{ij}\, c_{ij}(X) \geq \delta_{ij}\, c_{ij}(Y)$, $ij \neq u\,v$,

et par définition,

$$\delta_{uv}\, c_{uv}(X) = \delta_{uv}\, c_{uv}(Y) + f^{\delta}_{\mu+s}(Y)\ .$$

Donc $\sum_{\substack{i,j \\ i<j}} \delta_{ij}\, c_{ij}(X) \geq \sum_{\substack{i,j \\ i<j}} \delta_{ij}\, c_{ij}(Y) + f^{\delta}_{\mu+s}(Y)$.

La preuve procède par induction sur s .

Si $s = 1$, $F^{\delta}(X) \geq F^{\delta}(X^P) + f^{\delta}_{\mu+s}(X^P)$, par (***).

Et le théorème II est vrai par (*).

Supposons le théorème II vérifié pour $s - 1$ et démontrons-le pour s quelconque.

$$F^{\delta}(X^P) + \sum_{j \in L^P} \overline{x}_j \, f^{\delta}_j(X^P)$$

$$= F^{\delta}(X^P) + \sum_{j=\mu+1}^{\mu+s-1} \overline{x}_j \, f^{\delta}_j(X^P) + f^{\delta}_{\mu+s}(X^P) \text{ , par (*)}$$

$$= F^{\delta}(X^P) + \sum_{j=\mu+1}^{\mu+s-1} \overline{y}_j \, f^{\delta}_j(X^P) + f^{\delta}_{\mu+s}(X^P) \text{ , par (**)}$$

$$= F^{\delta}(X^P) + \sum_{j \in L^P} \overline{y}_j \, f^{\delta}_j(X^P) + f^{\delta}_{\mu+s}(X^P) \text{ , par (*,note) et par (**)}$$

$$\leq F^{\delta}(Y) + f^{\delta}_{\mu+s}(X^P) \text{ , par l'hypothèse d'induction}$$

$$\leq F^{\delta}(Y) + f^{\delta}_{\mu+s}(Y) \text{ , parce que } f^{\delta}_{\mu+s}(Y) \geq f^{\delta}_{\mu+s}(X^P)$$

En effet, lorsqu'on supprime l'arête $a_{\mu+s} = (u,v)$, ou bien la longueur de la plus courte chaîne entre u et v ne change pas ($f_{\mu+s} = 0$), ou bien la plus courte chaîne entre u et v est remplacée par une chaîne plus longue. Dans ce dernier cas, les possibilités de remplacement dans Y ne constituent qu'une partie des possibilités de remplacement dans X^P, et $f_{\mu+s}(Y) \geq f_{\mu+s}(X^P)$. Donc $f^{\delta}_{\mu+s}(Y) \geq f^{\delta}_{\mu+s}(X^P)$.

$\leq F^{\delta}(X)$, par (***).

C.Q.F.D.

Le théorème II garantit la validité de la borne inférieure I^P , ainsi redéfinie:

$$I^P = F^{\delta}(X^P) + \sum_{j \in L^P} \bar{x}_j^* \, f_j^{\delta}(X^P) \ ,$$

où $X^* = (x_1^*, x_2^*, \ldots, x_m^*)$ est la solution optimale du problème suivant:

$$\text{MIN} \quad F^{\delta}(X^P) + \sum_{j \in L^P} \bar{x}_j \, f_j^{\delta}(X^P)$$

en respectant les contraintes:

- $\sum_{j=\mu+1}^{m} c_j \, x_j \leq B$
- $x_j = 0$ si $-j$ apparaît dans la pile P
- $x_j = 1$ si j apparaît dans la pile P
- $0 \leq x_j \leq 1$, x_j réel , si $j \in L^P$.

5.4 Modifications à apporter à l'heuristique MQO

L'heuristique MQO est définie comme étant l'option 3 dans laquelle on utilise les Q_j à la place des f_j . Dans sa nouvelle version, l'heuristique MQO travaillera comme la nouvelle version de l'option 3 en employant les Q_j^{δ} à la place des f_j^{δ} .

CHAPITRE VI

CONCLUSION

Le problème du choix optimal d'un réseau (de transport) sans congestion consiste à déterminer parmi un ensemble de projets de construction de routes lesquelles de ces routes on va effectivement construire pour obtenir le meilleur réseau. Nous en avons fait l'objet de notre étude.

Notre travail a tout d'abord consisté à formuler et analyser le problème du point de vue théorique. Nous avons de plus étudié des algorithmes permettant de résoudre le problème. Enfin, nous avons fait l'évaluation numérique des algorithmes étudiés,à l'aide de programmes expérimentaux que nous avons écrits.

Nous avons partitionné la présentation de notre travail en six chapitres.

Dans un premier chapitre, nous avons passé en revue les travaux antérieurs apparaissant dans la littérature scientifique. Cela nous a permis d'établir le contexte de notre travail.

Dans un deuxième chapitre, nous avons étudié un algorithme de Murchland pour calculer les nouvelles longueurs des plus courts chemins après modification de la longueur d'un seul arc dans un réseau. Nous avons modifié l'algorithme de Murchland afin de le rendre encore plus efficace. Puis nous avons procédé à l'extension,au cas symétrique,de

l'algorithme de Murchland. Nous avons aussi donné une justification théorique ainsi qu'une évaluation numérique de tous les algorithmes dont nous venons de faire mention. Ce travail constituait un préalable au chapitre III.

Le chapitre III fait un compte rendu de notre recherche d'un algorithme efficace pour résoudre le problème du choix optimal d'un réseau. Dans cette recherche, nous avons combiné théorie et expérimentation. Le meilleur algorithme issu de cette recherche donne des résultats numériques que nous estimons être les meilleurs publiés jusqu'à maintenant.

Dans un quatrième chapitre, nous avons étudié quatre heuristiques. Parmi celles-ci, deux nouvelles heuristiques, peu coûteuses en temps de calcul et produisant une solution extrêmement voisine de la solution optimale.

Dans un cinquième chapitre, nous avons donné une formulation plus générale du problème du choix optimal d'un réseau et nous avons vu combien il était facile de modifier les algorithmes étudiés précédemment pour leur faire résoudre le problème dans sa formulation plus générale tout en augmentant très peu le temps de calcul.

Enfin, la présente conclusion constitue le chapitre VI.

Nous nous sommes limités dans la présente thèse à étudier le cas sans congestion. Un prolongement logique de notre travail pourrait se trouver dans l'étude du cas où il y a congestion. Lorsqu'on tient compte de la congestion, on obtient un problème nettement différent, en parti-

culier du point de vue numérique. Une autre étude intéressante serait celle de l'aspect dynamique du problème (...on construit certaines routes cette année, d'autres l'année suivante, d'autres encore l'année d'après, ...).

Une autre montagne sacrée a été gravie.
Zoom back camera.

Montréal, le 2 octobre 1974.

APPENDICE I

```
C-----------------------------------------------------------------
C
C
C     PROGRAMME OPTION
C     ----------------
C
C
C     UTILISATION :
C     -----------
C                    CE PROGRAMME EXPERIMENTAL PERMET DE RESOUDRE
C                    LE PROBLEME DU CHOIX OPTIMAL D'UN RESEAU
C                    SUIVANT LES ALGORITHMES OPTION1, 2, 3  OU 4
C                    DECRITS DANS LA THESE DE DOCTORAT DE L'AUTEUR
C                    (OCTOBRE 1974).
C                    IL A L'AVANTAGE DE CONTENIR AUSSI
C                    SOUS FORME DE SOUS-ROUTINES
C                    LES ALGORITHMES AS, DS  ET FLOYD97
C                    AINSI QUE LES HEURISTIQUES MGPM ET MOJ.
C
C
C
C     AUTEUR :       RENE DIONNE
C     ------                  CENTRE DE RECHERCHE SUR LES TRANSPORTS
C                             UNIVERSITE DE MONTREAL
C
C
C
C     LANGAGE :      FORTRAN EXTENDED  (FTN 4.0)
C     -------
C
C
C     ORDINATEUR :   CDC CYBER 74      UNIVERSITE DE MONTREAL
C     ----------
C
C
C-----------------------------------------------------------------

=
=
=
=
=
=
=
=
=
=
=
```

DESCRIPTION DES VARIABLES

PREMIEREMENT, CERTAINS NOMS DE VARIABLES SONT EXACTEMENT LES MEMES QUE CEUX EMPLOYES DANS LE CHAP. III DE LA THESE. AINSI EN EST-IL POUR

M	...NOMBRE D'ARETES
N	...NOMBRE DE SOMMETS
L(J)	...LONGUEUR DE L'ARETE A(J)
X	...VECTEUR XP I.E. X INDICE SUPERIEUR P (NOTE: X(J) PREND LA VALEUR -1 QUAND L'ARETE A(J) EST LIBRE)
X0	...VECTEUR X0
C	...MATRICE C
D	...MATRICE DES DISTANCES D
F(I)	...FONCTION PETIT F INDICE I
Q(I)	...FONCTION Q INDICE I
B	...BUDGET
IB	...INDIQUE POURCENTAGE DE LMAX
LMAX	...LONGUEUR TOTALE
LMIN	...LONGUEUR MINIMALE POUR UN RESEAU
DEG(I)	...DEGRE D'INCIDENCE AU SOMMET I
NIVD	...NIVEAU DANS L'ARBORESCENCE ASSOCIEE
NMAX	...NIVEAU MAXIMAL PERMIS
NEX	...NOMBRE D'ARETES EXCLUES
FREQ	...FREQUENCE D'EVALUATION DES F(J) OU Q(J)
PILE	...PILE P
NSOM	...NOMBRE DE SOMMETS DANS L'ARB. ASS.
NAS	...NOMBRE D'APPPELS DE L'ALGORITHME AS
NFLOYD	...NOMBRE D'APPELS DE L'ALGORITHME FLOYD97
NEVAL	...NOMBRE D'EVALUATIONS DES F(J) OU Q(J)
NBORNE	...NOMBRE D'EVALUATIONS DE LA BORNE IP
UTIL	...POURCENTAGE D'UTILITE DE LA BORNE
MCOUPE	...NIVEAU MOYEN DES COUPES
NOPT	...NUMERO DU SOMMET OPTIMAL DANS L'ARB.ASS.

```
1831=
1832=
1833=
1834=
1835=
1836=
1837=
1838=
1839=
1840=
1841=C        ----------------------------------------
1842=C         DESCRIPTION  DES  VARIABLES  (SUITE)
1843=C        ----------------------------------------
1844=C
1845=C
1846=C
1947=C           DEUXIEMEMENT, DES NOMS DE VARIABLES SONT PROPRES AU
1850=C        PROGRAMME :
1860=C
1870=C                    Z0      ...VALEUR DE F(X0)
1880=C                    LAMBDA0...COUT DE CONSTRUCTION DE X0
1890=C                    IFX     ...VALEUR DE F(X)
1900=C                    LAMBDA  ...COUT DE CONSTRUCTION DE X
1910=C                    ZBORNE  ...VALEUR DE LA BORNE INFERIEURE IP
1920=C                    VALANC  ...VECTEUR QUI CONTIENT LES ANCIENNES
1930=C                                       VALEURS DE VAL
1940=C                    NBORU   ...NOMBRE DE FOIS OU IP EST UTILE (COUPE)
1950=C                    PMCOUPE...MIS POUR MCOUPE=NIVEAU MOYEN DES COUPES
1960=C                    MEM1      /
1970=C                    MEM2    .../   SERT A CONSERVER DES ELEMENTS DE C
1980=C                    MEMC      /
1990=C                    COP     ...INDIQUE UNE COPIE
2000=C
2010=C
2020=C
2030=C
2040=C              ENFIN DES VARIABLES BOOLEENNES DEFINISSENT L'OPTION
2050=C        DE CALCUL :
2060=C
2070=C
2080=C
2090=C     •  LOGSG   = /.T.  SI LA SOLUTION INITIALE EST CALCULEE PAR MGPM
2100=C                  /.F.  SI LA SOLUTION INITIALE EST CALCULEE PAR MOJ
2110=C
2120=C     •  LOGSEV  = /.T. SIGNIFIE SANS EVALUATION ET SANS CALCUL DE IP
2130=C                  /.F. SIGNIFIE AVEC EVALUTION DES F(J) OU DES D(J)
2140=C
2150=C     •  LOGMUR  = /.T. SI LE CALCUL DES P.C.C. SE FAIT PAR L'ALG. AS
2160=C                  /.F. SI CE CALCUL  DES P.C.C. SE FAIT PAR FLOYD97
2170=C
2180=C     •  LOGF    = /.T.  SI ON CALCULE LES F(J)
2190=C                  /.F.  SI ON CALCULE LES D(J)
2200=C
2210=C     •  LOG1    = /.T.  SI LE BRANCHING SE FAIT SUR  X(1)
2220=C                  /.F.  SI CELUI-CI     SE FAIT SUR  X(P)
2230=
2240=
2250=
2260=
2270=
2280=
2290=
2300=
```

```
2540=
2550=
2560=
2570=
2580=
2590=
2600=
2610=
2620=
2630=C          ---------------------------------
2640=C           ENONCES  DU  PROGRAMME  OPTION
2650=C          ---------------------------------
2660=C
2670=C
2671=C
2672=C
2690=       PROGRAM   OPTION   (INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)

2790=C
2890=C
2990=C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
3090=C
3190=C            LE PROGRAMME PRINCIPAL GERE L'APPEL DES
3290=C          SOUS-ROUTINES AU NIVEAU LE PLUS PRIMITIF.
3390=C
3590=C  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
3600=C
3610=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
3620=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
3630=      -  D(30,30),C(30,30), RVAL(100), COP(30,30) ,
3640=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
3650=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,LMIN
3660=C
3670=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
3680=       LOGICAL      LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
3690=C  LECTURE ET IMPRESSION DES DONNEES DEFINISSANT LE RESEAU
3700=     1 READ(1,98) NCODE,INFINI
3710=       IF (EOF(1)) 2,3
3720=     2 STOP
3730=     3 READ (1,100) N,M
3740=       WRITE (2,101) NCODE,N,M
3750=       LMAX = 0
3760=       LMIN = INFINI
3770=       DO 5 I=1,M
3780=       READ (1,100) I1,I2,LL
3790=       IF (LL.LT.LMIN) LMIN = LL
3800=       LMAX = LMAX + LL
3810=       PT1(I) = I1    ;    PT2(I) = I2    ;    L(I) = LL
3820=     5 WRITE (2,100) I,I1,I2,LL
3830=C  LECTURE DES VARIABLES DE CONTROLE
3840=     7 READ (1,102) LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,IB

3850=       IF (EOF(1)) 1,8
3860=     8 B = LMAX * IB / 100
3870=       IF (.NOT.LOGSEV) GOTO 9
3880=       LOG1 = .T.
3890=     9 WRITE (2,103) LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,IB

3891=
3892=
3893=
3894=
3895=
3896=
```

```
3932=
3933=
3934=
3935=
3936=
3937=
3938=
3939=
3940=
3941=C  INITIALISATION
3942=       CALL SECOND(TIN)
3943=       IF (LOGSS) 10,12
3944=    10 CALL MGPM
3945=       GOTO 15
3950=    12 CALL MOJ
3960=C  IMPRESSION DE LA SOLUTION INITIALE
3970=    15 CALL SECOND(TOUT)
3980=       WRITE (2,99) LMAX
3990=       IF (Z0.EQ.INFINI) GOTO 7
4000=       TOUT = TOUT - TIN
4010=       WRITE (2,104) B
4020=       DO 18 I=1,M
4030=       IF (X0(I).EQ.0) GOTO 18
4040=       WRITE (2,105) I
4050=    18 CONTINUE
4060=       WRITE (2,106) Z0,LAMBDA0
4070=       WRITE (2,107) TOUT
4080=       IF (.NOT.LOGSEV) GOTO 25
4090=       DO 22 I=1,M
4100=    22 RANG(I) = I
4110=C
4120=C
4130=C    CALCUL ET IMPRESSION DE LA SOLUTION OPTIMALE
4140=C
4150=C
4160=    25 CALL SECOND(TIN)
4170=       CALL SEARCH
4180=       CALL SECOND(TOUT)
4190=       TOUT = TOUT - TIN
4200=       WRITE (2,108) B
4210=       DO 30 I=1,M
4220=       IF (X0(I).EQ.0) GOTO 30
4230=       WRITE (2,105) I
4240=    30 CONTINUE
4250=       WRITE (2,106) Z0,LAMBDA0
4260=       WRITE (2,109) TOUT
4270=       GOTO 7
4271=
4272=
4273=
4274=
4275=
4276=
4277=
4278=
4279=
4280=
4281=
4282=
4283=
4284=
4285=
4286=
```

```
4302=
4303=
4304=
4305=
4306=
4307=
4308=
4309=
4310=
4311=   98 FORMAT (A10,I10)
4312=   99 FORMAT (◆0     ..............LMAX = ◆,I10)
4313=  100 FORMAT (8I10)
4314=  101 FORMAT (      ◆1LE ◆,A10,◆ A◆,I2,◆ SOMMETS ET ◆,I3,◆ ARETES
ET IL
4320=     - EST DEFINI COMME SUIT : ◆ /  ◆        NUMERO    PT1       PT2

4330=     -  LONGUEUR ◆//)
4340=  102 FORMAT (5L2,3I10)
4350=  103 FORMAT (◆1DEFINITION DES VARIABLES DE CONTROLE :  LOGSG LOGSE
V LOG
4360=     -MUP  LOGF   LOG1   FREQ   NMAX     IP◆//,40X,5(L3,4X),I5,2I8)

4370=  104 FORMAT (///◆ POUR UN BUDGET DE ◆,I5,/◆ LA SOLUTION INITIALE R
ETIEN
4380=     -T LES ARETES SUIVANTES :◆)
4390=  105 FORMAT (  40X,◆ARETE◆,I4)
4400=  106 FORMAT (20X,◆Z0 =◆,I7,10X,◆LAMBDA0 =◆,I7)
4410=  107 FORMAT (/,◆ TEMPS DE CALCUL DE LA SOLUTION INITIALE : ◆,F10.3
,◆
4420=     -SEC.◆)
4430=  108 FORMAT (    ◆1POUR UN BUDGET DE ◆,I5,/◆ LA SOLUTION OPTIMALE R
ETIEN
4440=     -T LES ARETES SUIVANTES :◆)
4450=  109 FORMAT (/,◆ TEMPS DE CALCUL DE LA SOLUTION OPTIMALE : ◆,F10.3
,◆
4460=     -SEC.◆)
4470=      END
4471=
4472=
4473=
4474=
4475=
4476=
4477=
4478=
4479=
4480=
4481=
4482=
4483=
4484=
4485=
4486=
4487=
4488=
4489=
4490=
4491=
4492=
4493=
4494=
4495=
4496=
4497=
```

```
4513=
4514=
4515=
4516=
4517=
4518=
4519=
4520=
4521=       SUBROUTINE  MGPM
4522=C
4523=C                 ---------------------
4524=C                   HEURISTIQUE MGPM
4525=C                 ---------------------
4526=C
4527=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
4528=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
4529=      -  D(30,30),C(30,30), RVAL(100), COP(30,30),
4530=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
4531=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,LMIN
4540=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
4550=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
4560=       INTEGER PILE(30)
4570=       DO 5 I=1,N
4580=       DO 5 J=1,N
4590=     5 D(I,J) = 0
4600=C  ON ORDONNE LES ARETES SUIVANT L
4610=       CALL ORDRE(L,1,M)
4620=C  ON CONSTRUIT LE GRAPHE PARTIEL CONNEXE MIN  (MIN SPANNING TREE)

4630=C  NAPLAC :  NOMBRE D'ARETES PLACEES DANS L'ARBRE
4640=C  NARESS :  NOMBRE D'ARETES ESSAYEES ...
4650=C  LE VECTEUR X SERT D'ETIQUETTE . X(I)= 1  SIGNIFIE QUE LE SOMMET
I EST MARQUE
4660=       NAPLAC = NARESS = 0
4670=       DO 11 I=1,M
4680=    11 X0(I) = 0
4690=    12 NARESS = NARESS + 1
4700=       IF (NARESS.GT.M) GOTO 30
4710=       K = RANG(NARESS)
4720=       I1 = PT1(K)    $    I2 = PT2(K)    $    LL = L(K)
4721=
4722=
4723=
4724=
4725=
4726=
4727=
4728=
4729=
4730=
4731=
4732=
4733=
4734=
4735=
4736=
4737=
4738=
4739=
4740=
4741=
4742=
4743=
4744=
```

```
4766=
4767=
4768=
4769=
4770=
4771=       GOTO 30
4772=C  POUR DETERMINER SI L'ADDITION DE L'ARETE CREE UN CYCLE
4773=C       RETOUR A 15 S'IL N'Y EN A PAS.  SINON RETOUR A 12
4774=    15 D(I1,I2) = D(I2,I1) = LL
4775=       NAPLAC = NAPLAC + 1
4780=       X0(K) = 1
4790=       IF (NAPLAC.EQ.N-1) 38,12
4800=    30 DO 22 I=1,N
4810=    22 X(I) = 0
4820=       X(I1) = 1      $    X(I2) = 1
4830=       IEXCL = I1     $    IPILE = I2     $    IDEBUT = 0
4840=       PILE(1) = I1   $    PILE(2) = I2
4850=       NIVO = 2
4860=    25 IDEBUT = IDEBUT + 1
4870=       IF (IDEBUT.GT.N) GOTO 35
4880=       IF (D(IPILE,IDEBUT).EQ.0) GOTO 25
4890=       IF (IDEBUT.EQ.IEXCL) GOTO 25
4900=       IF (X(IDEBUT).EQ.1) GOTO 18
4910=       X(IDEBUT) = 1
4920=       IEXCL = IPILE
4930=       IPILE = IDEBUT
4940=       NIVO = NIVO + 1
4950=       PILE(NIVO) = IPILE
4960=       IDEBUT = 0
4970=       GOTO 25
4980=    35 X(IPILE) = 0
4990=       NIVO = NIVO - 1
5000=       IF(NIVO.EQ.1) GOTO 15
5010=       IDEBUT = IPILE
5020=       IPILE = IEXCL
5030=       IEXCL = PILE(NIVO-1)
5040=       GOTO 25
5050=    38 WRITE (2,100) B
5060=       Z0 = INFINI
5070=       RETURN
5080=    39 LONG = 0
5090=       DO 40 I=1,M
5100=       IF (X0(I).EQ.1)  LONG = LONG + L(I)
5110=    40 CONTINUE
5120=       IF(LONG.GT.B) GOTO 38
5130=       WRITE (2,999) LONG
5140=       DO 45 I=1,M
5150=       K = RANG(I)
5160=       IF (X0(K).EQ.1) GOTO 45
5170=       LL = LONG +L(K)
5180=       IF (LL.GT.B) GOTO 50
5190=       X0(K) = 1
5200=       LONG = LL
5241=    45 CONTINUE
5242=    50 LAMBDA0 = LONG - B
5243=       CALL FLOYD(X0,Z0)
5244=   100 FORMAT (/' *  AUCUN GRAPHE PARTIEL CONNEXE REAL.  ...B =*,I8)
5250=   999 FORMAT (/' *  ........ LMIN = *,I5)
5260=       RETURN
5270=       END
5271=
5272=
5273=
5274=
```

```
5292=
5293=
5294=
5295=
5296=
5297=
5298=
5299=
5300=
5301=       SUBROUTINE ORDRE(LACLE,INF,ISUP)
5302=C
5303=C  ..... ... .....
5304=C  CETTE SOUS-ROUTINE INDIQUE QUEL EST L'ORDRE CROISSANT DANS
5305=C  LA PARTIE DU VECTEUR LACLE QUI VA DE LACLE(INF) A LACLE(ISUP)
5306=C  LE TRAVAIL SE FAIT DANS LE VECTEUR LEVT SANS MODIFIER LE VECTEUR
5307=C  LACLE       ...ET A LA FIN, RANG(I) INDIQUE LA POSITION
5308=C  ELEMENT DU VECTEUR LACLE.
5309=C  .....
5310=C
5311=C
5312=C
5313=       COMMON  M,N,P,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
5314=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
5315=      -  D(30,30),C(30,30), PVAL(100), COP(30,30) ,
5320=      -  LOGSG,LOGSEV,LOGMUP,LOGF,LOG1,FREQ,NMAX,
5330=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NROBU,IMIN
5340=       INTEGER P,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
5350=       LOGICAL      LOGSG,LOGSEV,LOGMUP,LOGF,LOG1
5360=       DIMENSION LACLE(1),LEVT(100)
5361=C
5362=C
5410=       IF (INF.EQ.ISUP) 1,2
5420=     1 RANG(INF) = INF
5430=       RETURN
5440=     2 DO 5 I=INF,ISUP
5450=       RANG(I) = I
5460=     5 LEVT(I) = LACLE(I)
5470=       IM1 = ISUP-1
5480=       DO 10 I=INF,IM1
5490=       IP1 = I+1
5500=       K = I
5510=       DO 7 J=IP1,ISUP
5520=       IF (LEVT(J).LT.LEVT(K)) K=J
5530=     7 CONTINUE
5540=       NPERM = RANG(I)  $  RANG(I) = RANG(K)  $  RANG(K) = NPERM

5550=       NPERM = LEVT(I)  $  LEVT(I) = LEVT(K)  $  LEVT(K) = NPERM

5560=    10 CONTINUE
5570=       RETURN
5580=       END
5581=
5582=
5583=
5584=
5585=
5586=
5587=
5588=
5589=
5590=
5591=
5592=
5593=
```

```
5623=
5624=
5625=
5626=
5627=
5628=
5629=
5630=
5631=      SUBROUTINE FLOYD(V,IZ)
5632=      INTEGER V(1)
5633=      COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
5634=     -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
5635=     -  D(30,30),C(30,30), RVAL(100), COP(30,30) ,
5640=     -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
5650=     - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORI,LMIN
5660=      INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL ,D,C,FREQ,COP
5670=      LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
5671=C
5680=C  INITIALISATION
5681=C
5690=      DO 2 I=1,N
5700=      DO 2 J=1,N
5710=    2 C(I,J) = INFINI
5720=      DO 5 I=1,M
5730=      IF (V(I)) 4,5,4
5740=    4 I1 = PT1(I)  $   I2 = PT2(I)
5750=      C(I1,I2) = C(I2,I1) = L(I)
5760=    5 CONTINUE
5770=      DO 6 I=1,N
5780=    6 C(I,I) = 0
5781=C
5790=C  ALGORITHME FLOYD97
5791=C              --------
5792=C
5800=      NM1 = N-1
5810=      DO 10 K=1,N
5820=      DO  9 I=1,NM1
5830=      IF (C(I,K).GE.INFINI) GOTO 9
5840=      IP1 = I + 1
5850=      DO  8 J=IP1,N
5860=      IF (C(K,J).GE.INFINI) GOTO 8
5870=      LL = C(I,K) + C(K,J)
5880=      IF ( C(I,J).LE.LL)  GOTO 8
5890=      C(I,J) = C(J,I) = LL
5900=    8 CONTINUE
5910=    9 CONTINUE
5920=   10 CONTINUE
5921=C
5930=C  SOMME  DES  PLUS COURTES CHAINES
5931=C
5940=      IZ = 0
5950=      DO 15 I=1,NM1
5960=      IP1 = I+1
5970=      DO 15 J=IP1,N
5980=   15 IZ = IZ + C(I,J)                                    ...
5990=      RETURN
6000=      END
6001=
6002=
6003=
6004=
6005=
6006=
6007=
```

```
6032=
6033=
6034=
6035=
6036=
6037=
6038=
6039=
6040=
6041=       SUBROUTINE  MGJ
6042=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
6043=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),PANG(100),VAL(100),
6044=      -  D(30,30),C(30,30), PVAL(100), COP(30,30) ,
6050=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
6060=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NROPT,LMIN
6070=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,PANG,VAL ,D,C,FREQ,COP
6080=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
6090=       INTEGER O,G
6100=       DO 1 I=1,N
6110=       DO 1 K=1,N
6120=     1 D(I,K) = INFINI
6130=       DO 2 I=1,M
6140=       IF (I.LE.N) D(I,I)=0
6150=       I1 = PT1(I)    $    I2 = PT2(I)
6160=       D(I1,I2) = D(I2,I1) = L(I)
6170=     2 X(I) = 1
6180=       CALL FLOYD(X,Z0)
6190=       LAMBDA0 = 0
6200=       DO 5 I=1,M
6210=     5 LAMBDA0 = LAMBDA0 + L(I)
6220=       LAMBDA0 = LAMBDA0 - B
6230=     6 DO 10 I=1,M
6240=       IF (X(I).EQ.0) GOTO 8
6250=       VAL(I) = D(I)
6260=       GOTO 10
6270=     8 VAL(I) = INFINI
6280=    10 CONTINUE
6290=       K=1
6300=       DO 12 I=1,M
6310=       IF(VAL(I).LT.VAL(K))  K=I
6320=    12 CONTINUE
6330=       IF (VAL(K).EQ.INFINI) 15,16
6340=    15 CALL MGFM
6350=       RETURN
6360=    16 X(K) = 0
6370=       LAMBDA0 = LAMBDA0 - L(K)
6380=       Z0 = Z0 + VAL(K)
6390=       I1 = PT1(K)    $    I2 = PT2(K)    $    L = L(K)
6400=       D(I1,I2) = D(I2,I1) = INFINI
6410=       CALL AS(I1,I2,LL)
6420=    18 IF (LAMBDA0) 20,40,6
6430=    20 K=0
6440=       DO 25 I=1,M
6450=       IF (X(I).NE.0) GOTO 25
6460=       IF(LAMBDA0+L(I).GT.0) GOTO 25
6470=       K = K+1
6480=       PILE(K) = I
6490=    25 CONTINUE
6491=
6492=
6493=
6494=
6495=
6496=
```

```
6520=
6521=
6522=
6523=
6524=
6525=
6526=
6527=
6528=
6529=
6530=
6531=       IF(K-1) 40,28,30
6532=    28 I = PILE(K)
6533=       X(I) = 1
6534=       LAMBDA0 = LAMBDA0 + L(I)
6540=       Z0 = Z0 - G(I)
6550=       GOTO 40
6560=    30 IMAX = 1    $    IVAL = 0
6570=       DO 35 I=1,K
6580=       II = PILE(I)
6590=       IG = G(II)
6600=       IF (IVAL.LT.IG) 32,35
6610=    32 IVAL = IG
6620=       IMAX = II
6630=    35 CONTINUE
6640=       X(IMAX) = 1
6650=       Z0 = Z0 - IVAL
6660=       LAMBDA0 = LAMBDA0 + L(IMAX)
6670=       I1 = PT1(IMAX)    $    I2 = PT2(IMAX)    $    LL = L(IMAX)

6680=       CALL DS(I1,I2,LL)
6690=       D(I1,I2) = D(I2,I1) = LL
6700=       GOTO 18
6710=    40 DO 42 I=1,M
6720=    42 X0(I) = X(I)
6730=       RETURN
6740=       END
6741=
6742=
6743=
6744=
6745=
6746=
6747=
6748=
6749=
6750=
6751=
6752=
6753=
6754=
6755=
6756=
6757=
6758=
6759=
6760=
6761=
6762=
6763=
6764=
6765=
```

```
6786=
6787=
6788=
6789=
6790=
6791=
6792=
6793=
6794=
6795=
6796=       SUBROUTINE DS(J,K,DPRIME)
6797=C
6798=C                      -------------
6799=C                      ALGORITHME DS
6800=C                      -------------
6801=C
6802=C
6803=       INTEGER DPRIME,IPILE(30),LPILE(30)
6804=       COMMON  M,N,S,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
6805=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
6806=      -  D(30,30),C(30,30), RVAL(100), CDP(30,30) ,
6807=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
6810=     - NSOM,NAS,NFLOYD,NFVAL,NBORNE,NBOPU,LMIN
6820=       INTEGER S,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL ,D,C,FREQ,CDP
6830=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
6840=
6850=C    ETAPE DS1
6860=     5 NI = NL = 1
6870=C    ETAPE DS2
6880=       DO 10 I=1,N
6890=       MI = C(I,J) + DPRIME
6900=       IF ( C(I,K).LE.MI) GOTO 10
6910=       IPILE(NI) = I    ;    NI = NI + 1
6920=       C(I,K) = C(K,I) = MI
6930=    10 CONTINUE
6940=C    ETAPE DS3
6950=       DO 20 LL=1,N
6960=       ML = DPRIME + C(K,LL)
6970=       IF (C(J,LL).LE.ML) GOTO 20
6980=       LPILE(NL) = LL   ;    NL = NL + 1
6990=       C(J,LL)= C(LL,J) = ML
7000=    20 CONTINUE
7010=C    ETAPE DS4
702 =    25 ML = NL
7030=       NI = NI - 1
7040=       IF(NI.EQ.0) GOTO 40
7050=      I = IPILE(NI)
7060=    30 ML = ML - 1
7070=       IF (ML.EQ.0) GOTO 25
7080=      LL = LPILE(ML)
7090=       MS = C(I,J) + C(J,LL)
7100=       IF (C(I,LL).LE.MS) GOTO 30
7110=      C(I,LL) = C(LL,I) = MS
7120=      GOTO 30
7130=    40 RETURN
7140=       END
7141=
7142=
7143=
7144=
7145=
7146=
7147=
7148=
```

```
7160=
7161=
7162=
7163=
7164=
7165=
7166=       INTEGER FUNCTION O(NN)
7167=       INTEGER DJK,CINF,IPILE(30),LPILE(30),PILE1(300),PILE2(300)

7170=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
7180=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
7190=      -  D(30,30),C(30,30), RVAL(1 0), COP(30,30) ,
7200=      -  LOGSG,LOGSEV,LOGMUP,LOGF,LOG1,FREQ,NMAX,
7210=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBOPU,LMIN
7220=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
7230=       LOGICAL       LOGSG,LOGSEV,LOGMUP,LOGF,LOG1
7240=
7250=       J = PT1(NN)    $    K = PT2(NN)    $    DJK =L(NN)
7260=       NM1 = N-1
7270=
7280=       I0 = 0
7290=      IF (C(J,K).LT.DJK) GOTO 191
7300=       DO10 I=1,N
7310=       DO 10 LL=1,N
7320=    10 COP(I,LL) = C(I,LL)
7330=       D(J,K) = D(K,J) = INFINI
7340=
7350=C   ETAPE AS1
7360=       NI = NL = 1
7370=C   ETAPE AS2
7380=       DO 40 I=1,N
7390=       MS = COP(I,J) + DJK
7400=       IF (COP(I,K).NE.MS) GOTO 40
7410=      IPILE(NI) = I
7420=       NI = NI + 1
7430=    40 CONTINUE
7440=C    ETAPE AS3
7450=       DO 50 LL=1,N
7460=       MS = DJK + COP(K,LL)
7470=      IF (COP(J,LL).NE.MS) GOTO 50
7480=       LPILE(NL) = LL
7490=       NL = NL + 1
7500=    50 CONTINUE
7510=C    ETAPE AS4
7520=       MPILE = 1
7530=    60 ML = NL
7540=      NI = NI - 1
7550=       IF (NI.EQ.0) GOTO 80
7560=       I = IPILE(NI)
7570=    70 ML = ML - 1
7580=       IF (ML.EQ.0) GOTO 60
7590=       LL = LPILE(ML)
7600=      MS = COP(I,J) + DJK + COP(K,LL)
7610=      IF ( COP(I,LL).NE.MS) GOTO 70
7620=      COP(I,LL) = COP(LL,I) = D(I,LL)
7630=       IF (MPILE.GT.300) GOTO 92
7640=       PILE1(MPILE) = I
7650=      PILE2(MPILE) = LL
7660=      MPILE = MPILE + 1
7670=       GOTO 70
7680=    80 N12 = MPILE - 1
7681=
7682=
7683=
```

```
7713=
7714=
7715=
7716=
7717=
7718=
7719=
7720=
7721=C    ETAPE A35
7722=      DO 30 MM=1,N
7723=        DO90 MPILE = 1,N12
7724=       I =PILE1(MPILE)
7730=        LL = PILE2(MPILE)
7740=        MS = COP(I,MM)+ COP(MM,LL)
7750=       IF ( COP(I,LL).LE.MS )  GOTO 90
7760=        COP(I,LL) = COP(LL,I) = MS
7770=     90 CONTINUE
7790=
7800=       DO 91 I=1,NM1
7810=        IP1 = I+1
782 =       DO 91 LL = AP1,N
7830=        CINF =COP(I,LL)
7840=        IF (CINF.GE.INFINI) GOTO 92
7850=     91 IO = IO + CINF - C(I,LL)
7860=   1910 = IO
7870=        D(J,K) = D(K,J) = DJK
7880=      RETURN
7890=     92 WRITE(2,100)
7900=        RETURN
7910=   93 O = INFINI
792 =      RETURN
7930=   100 FORMAT (♦          ......DEPASSEMENT DE CAPACITE POUR PILE1 ET 2
♦)
7940=        END
7941=
7942=
7943=
7944=
7945=
7946=
7947=
7948=
7949=
7950=
7951=
7952=
7953=
7955=
7956=
7957=
7958=
7959=
7960=
7961=
7962=
7963=
7964=
7965=
7966=
7967=
7968=
7969=
7170=
7971=
```

```
7992=
7993=
7994=
7995=
7996=
7997=
7998=
7999=
8000=
8001=       SUBROUTINE AS(J,K,DJK)
8002=C
8003=C                  ---------------
8004=C                   ALGORITHME AS
8005=C                  ---------------
8006=C
8007=       INTEGER DJK,IPILE(30),LPILE(30),PILE1(300),PILE2(300)
8008=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
8009=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
8010=      -  D(30,30),C(30,30), PVAL(100), COP(30,30) ,
8011=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
8012=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,LMIN
8020=       INTEGER B,PT1,PT2, N,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
8030=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
8040=
8050=C    ETAPE AS1
8060=       NI = NL = 1
8070=C    ETAPE AS2
8080=       DO 40 I=1,N
8090=       MS = C(I,J) + DJK
8100=       IF (C(I,K).NE.MS) GOTO 40
8110=       IPILE(NI) = I
8120=       NI = NI + 1
9130=    40 CONTINUE
8140=C    ETAPE AS3
8150=       DO 50 LL=1,N
8160=       MS = DJK + C(K,LL)
8170=       IF (C(J,LL).NE.MS) GOTO 50
8180=       LPILE(NL) = LL
8190=       NL = NL + 1
8200=    50 CONTINUE
8210=C    ETAPE AS4
8220=       MPILE = 1
8230=    60 ML = NL
8240=       NI = NI - 1
8250=       IF (NI.EQ.0) GOTO 80
8260=       I = IPILE(NI)
8270=    70 ML = ML - 1
8280=       IF (ML.EQ.0) GOTO 60
8290=       LL = LPILE(ML)
8300=       MS = C(I,J) + DJK + C(K,LL)
8310=       IF ( C(I,LL).NE.MS) GOTO 70
8320=       C(I,LL) = C(LL,I) = D(I,LL)
8330=       IF (MPILE.GT.300) GOTO 90
8340=       PILE1(MPILE) = I
8350=       PILE2(MPILE) = LL
8360=       MPILE = MPILE + 1
8370=       GOTO 70
8380=    80 N12 = MPILE - 1
8381=
8382=
8383=
8384=
8385=
8386=
```

```
8402=
8403=
8404=
8405=
8406=
8407=
8408=
8409=
8410=
8411=C    ETAPE AS5
8412=        DO 90 MM=1,N
8413=        DO 90 MPILE = 1,N12
8420=        I = PILE1(MPILE)
8430=        LL = PILE2(MPILE)
8440=        MS = C(I,MM) + C(MM,LL)
8450=        IF ( C(I,LL).LE.MS )  GOTO 90
8460=        C(I,LL) = C(LL,I) = MS
8470=     90 CONTINUE
8480=        RETURN
8490=
8500=     92 WRITE(2,100)
8510=        RETURN
8520=    100 FORMAT (♦     ..... DEPASSEMENT DE CAPACITE POUR PILE1 ET 2
♦)
8530=        END
8531=
8532=
8533=
8534=
8535=
8536=
8537=
8538=
8539=
8540=
8541=
8542=
8543=
8544=
8545=
8546=
8547=
8548=
8549=
8550=
8551=
8552=
8553=
8554=
8555=
8556=
8557=
8558=
8559=
8560=
8561=
8562=
8563=
8564=
8565=
8566=
8567=
8568=
8569=
```

```
8582=
8583=
8584=
8585=
8586=
8587=
8588=
8589=
8590=
8591=       INTEGER FUNCTION G(NN)
8592=C
8593=C      ..................................................................
8594=C      CETTE SOUS-ROUTINE CALCULE LA DIMINUTION DE LA FONCTION ECON.
8595=C      CAUSEE PAR L'ADDITION DE L'ARETE A(NN).
8596=C      ..................................................................

8597=       INTEGER DPRIME,IPILE(30),LPILE(30)
8598=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
8599=      -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
8600=      -  D(30,30),C(30,30), RVAL(100), COP(30,30),
8601=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,EPEQ,NMAX,
8602=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBOPU,LMIN
8610=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,EPEQ,COP
8620=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
8630=
8640=       J = PT1(NN)    $    K = PT2(NN)    $    DPRIME = L(NN)
8650=
8660=       IG = 0
8670=       IF (C(J,K).LE.DPRIME) GOTO 40
8680=       DO 5 I=1,N
8690=       DO 5 LL=1,N
8700=     5 COP(I,LL) = C(I,LL)
8710=
8720=C    ETAPE DS1
8730=       NI = NL = 1
8740=C    ETAPE DS2
8750=       DO 10 I=1,N
8760=       MI = COP(I,J) + DPRIME
8770=       IF ( COP(I,K).LE.MI) GOTO 10
8780=       IPILE(NI) = I    $    NI = NI + 1
8790=       IG = IG + COP(I,K) - MI
8800=       COP(I,K) = COP(K,I) = MI
8810=    10 CONTINUE
8820=C    ETAPE DS3
8830=       DO 20 LL=1,N
8840=       ML = DPRIME + COP(K,LL)
8850=       IF (COP(J,LL).LE.ML) GOTO 20
8860=       LPILE(NL) = LL   $    NL = NL + 1
8870=       IG = IG + COP(J,LL) - ML
8880=       COP(J,LL) = COP(LL,J) = MI
8890=    20 CONTINUE
8891=
8892=
8893=
8894=
8895=
8896=
8897=
8898=
8899=
8900=
8901=
9902=
8903=
```

```
8916=
8917=
8918=
8919=
8920=
8921=C    ETAPE DS4
8922=     25 ML = NL
8923=        NI = NI - 1
8930=        IF(NI.EQ.0) GOTO 40
8940=        I = IPILE(NI)
8950=     30 ML = ML - 1
8960=        IF(ML.EQ.0) GOTO 25
8970=        LL = LPILE(ML)
8980=        MS = COP(I,J) + COP(J,LL)
8990=        IF (COP(I,LL).LE.MS) GOTO 30
9000=        IG = IG + COP(I,LL) - MS
9010=        COP(I,LL) = COP(LL,I) = MS
9020=        GOTO 30
9030=     40 G = IG
9040=        RETURN
9050=        END
9051=C
9052=
9053=
9054=
9055=
9056=
9057=C    ******************************************************************
9060=
9061=
9062=
9063=
9064=
9065=
9066=
9067=
9068=
9069=
9070=
9071=        FUNCTION IDIJK(IS,IT)
9072=C
9073=C        ..................................................
9074=C        CETTE SOUS-ROUTINE CALCULE LA LONGUEUR DE LA PLUS COURTE
9075=C        CHAINE ENTRE IS ET IT SUIVANT L'ALGORITHME DE DIJKSTRA.
9076=C        ..................................................
9077=C
9078=        COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
9080=       -  LAMBDA0,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
9090=       -  D(30,30),C(30,30), RVAL(100), COP(20,20) ,
9100=       -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
9110=       - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,LMIN
9120=        INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
9130=        LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
9140=        LOGICAL TEST
9150=        DIMENSION NSET(50),LSET(50),LETIQ(30)
9160=        NDEP = IS
9170=        NB = 0
9180=        DO 10 I=1,N
9190=     10 LETIQ(I) = INFINI
9200=        LETIQ(NDEP) = 0
9201=
9202=
9203=
9204=
```

```
9226=
9227=
9228=
9229=
9230=
9231=
9232=
9233=
9234=
9235=
9236=
9237=    12 DO 20 I=1,N
9238=          ND = D(NDER,I)
9240=          IF (ND.EQ.INFINI) GOTO 20
9250=          IF ( NDER.EQ.I   ) GOTO 20
9260=          IF (LETIQ(I) .EQ.INFINI) 15,20
9270=    15    NB = NB + 1
9280=          IF (NB.GT.50) GOTO 30
9290=          NSET(NB) = I
9300=          LSET(NB) = LETIQ(NDER) + ND
9310=    20 CONTINUE
9320=
9330=    25 IF (NB.EQ.0) GOTO 60
9340=          LMIN = INFINI
9350=          DO 30 I=1,NB
9360=          IF (LMIN.LE.LSET(I)) GOTO 30
9370=          LMIN = LSET(I)    $    K = I
9380=    30 CONTINUE
9390=
9400=          TEST = .F.
9410=          NDER = NSET(K)
9420=          IF (LETIQ(NDER).LT.INFINI) GOTO 35
9430=          LETIQ(NDER) = LMIN
9440=          IF (NDER.EQ.IT) GOTO 50
9450=          TEST = .T.
9460=
9470=    35    NB = NB - 1
9480=          IF (NB.EQ.0) GOTO 41
9490=          DO 40 I=K,NB
9500=          IP1 = I + 1
9510=          NSET(I) = NSET(IP1)
9520=    40    LSET(I) = LSET(IP1)
9530=    41    IF (TEST) 10,25
9540=
9550=    50 IDIJK = LETIQ(IT)    $    RETURN
9560=    60 IDIJK = INFINI       $    RETURN
9570=    90 WRITE (2,100)
9580=   100 FORMAT (*  DEPASSEMENT DE CAPACITE :  NB    *)
9590=          END
9591=
9592=
9593=
9594=
9595=
9596=
9597=
9598=
9599=
9600=
9601=
9602=
9603=
9604=
9605=
```

```
9607=
9608=
9609=
9610=
9611=
9612=
9613=
9614=
9615=
9616=       SUBROUTINE SEARCH
9617=C
9618=C     ..................................  ...
9619=C     ......... OPTION 1, 2, 3, 4 ........  .
9620=C     ....................................
9621=C
9622=C
9623=C    COMMON
9630=       COMMON  M,N,B,INFINI,PT1(100),PT2(100),L(100),IND,ZBORNE,
9640=      -  LAMBDA,Z0,X(100),X0(100),PILE(100),RANG(100),VAL(100),
9650=      -  D(30,30),C(30,30), RVAL(100), COP(30,30) ,
9660=      -  LOGSG,LOGSEV,LOGMUR,LOGF,LOG1,FREQ,NMAX,
9670=      - NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,LMIN
9680=       INTEGER B,PT1,PT2, X,X0 ,Z0,PILE,RANG,VAL,D,C,FREQ,COP
9690=       LOGICAL       LOGSG,LOGSEV,LOGMUR,LOGF,LOG1
9700=C
9710=       INTEGER VALANC(2000),NDE(100),NA(100),MEM1(2000),MEM2(2000),

9720=      -MEMC(2000),DEG(30)
9730=       INTEGER O
9740=       REAL OVALANC(2000),OVAL(100)
9750=C
9760=C    INITIALISATION
9770=C
9780=       NSOM = NAS = NFLOYD = 0
9790=       NEVAL = NBORNE = NBORU = MCOUPE = 0
9800=       NIVO = NEL = 0
9810=       NDEP = NOPT = 1
9820=       DO 5 I=1,N
9830=       DO 6 J=1,N
9840=     6 D(I,J) = INFINI
9850=       DEG(I) = 0
9860=     5 D(I,I) = 0
9870=       LAMBDA = 0
9880=       DO 10 I=1,M
9890=       I1 = PT1(I)    $    I2 = PT2(I)     $     I3 = L(I)
9900=       LAMBDA = LAMBDA + I3
9910=       DEG(I1) = DEG(I1) + 1
9920=       DEG(I2) = DEG(I2) + 1
9930=       D(I1,I2) = D(I2,I1) = I3
9940=    10 X(I) = -1
9950=       LAMBDA = LAMBDA - B
9960=       CALL FLOYD(X,IFX)
9970=       NFLOYD = NFLOYD + 1
9980=C
9981=
9982=
9983=
9984=
9985=
9986=
9987=
9988=
9989=
9990=
```

```
10001=
10002=
10003=
10004=
10005=
10006=
10007=
10008=
10009=
10010=
10011=C    SEARCH
10012=C
10013=    20 IF (NIVO.EQ.M) GOTO 400
10020=       IF (.NOT.LOGSEV) GOTO 50
10030=       IND = NIVO + 1
10040=       GOTO 200
10050=    50 IF (NIVO.EQ.0) GOTO 60
10060=       IND = PILE(NIVO)
10070=       IF (IND.GT.0) GOTO 150
10080=       IMOD = NEL - (NEL/FREQ)*FREQ
10090=       IF (IMOD.NE.0) GOTO 150
10100=       IF (NIVO.GT.NMAX) GOTO 150
10110=    60 IF (LOGMUR) GOTO 110
10120=       CALL FLOYD(X,IFX)
10130=       NFLOYD = NFLOYD + 1
10140=       NIV = (NEL/FREQ) + 1
10150=       MEMC(NIV) = IFX
10160=   100 IF (IFX.GE.ZO) GOTO 400
10170=   110 NEVAL = NEVAL + 1
10180=       IF (LOGF) 130,120
10190=C
10200=C    CALCUL DES Q
10210=C
10220=   120 IF (.NOT.LOGMUR) GOTO 126
10230=       DO 125 I=1,M
10240=       IF (X(I).EQ.-1) 122,121
10250=   121 QVAL(I) = INFINI
10260=       GOTO 125
10270=   122 TVAL = Q(I)
10280=       QVAL(I) = TVAL / L(I)
10290=   125 CONTINUE
10300=       GOTO 130
10310=   126 CALL FLOYD(X,IW)
10320=       DO 129 I=1,M
10330=       IF (X(I).EQ.-1) 127,128
10340=   127 X(I) = 0
10350=       CALL FLOYD(X,IWO)
10360=       X(I) = -1
10370=       TVAL = IWO - IW
10380=       QVAL(I) = TVAL / L(I)
10390=       GOTO 129
10400=   128 QVAL(I) = INFINI
10410=   129 CONTINUE
10420=C
10421=
10422=
10423=
10424=
10425=
10426=
10427=
10428=
10429=
10430=
```

```
10433=
10434=
10435=
10436=
10437=
10438=
10439=
10440=
10441=C    CALCUL DES F
10442=C
10450=   130 DO 136 I=1,M
10460=       IF (X(I).EQ.-1) 132,131
10470=   131 VAL(I) = INFINI
10480=       GOTO 136
10490=   132 I1 = PT1(I)    $    I2 = PT2(I)
10500=       D(I1,I2) = D(I2,I1) = INFINI
10510=       NTEMP = IDIJK(I1,I2)
10520=       D(I1,I2) = D(I2,I1) = L(I)
10530=       IF (NTEMP.EQ.INFINI) 133,134
10540=   133 VAL(I) = INFINI
10550=       GOTO 136
10560=   134 VAL(I) = NTEMP - IDIJK(I1,I2)
10570=   136 CONTINUE
10580=C
10590=C    EMMAGASINER LES VAL(I)
10600=C
10610=   140 NDEB1 = (NEL/FREQ) * M
10620=       IF (NDEB1+M.GT.2000) GOTO 500
10630=       DO 145 I=1,M
10640=        ND = NDEB1+I
10650=   145 VALANC(ND) =  VAL(I)
10660=       IF (LOGF) GOTO 150
10670=       DO 146 I=1,M
10680=        ND = NDEB1+I
10690=   146  OVALANC(ND) = OVAL(I)
10700=C
10710=C    KNAPSACK FRACTIONNAIRE
10720=C
10730=   150 DO 153 I=1,M
10740=       TVAL = VAL(I)
10750=       RVAL(I) = TVAL / L(I)
10760=   153 RANG(I) = I
10770=       CALL ORDRE(RVAL,1,M)
10780=       ZBORNE = IFX
10790=       NBORNE = NBORNE + 1
10800=       LRESTE = LAMBDA
10810=       DO 159 I=1,M
10820=       IR = RANG(I)
10830=       IF (VAL(IR).EQ.INFINI) GOTO 999
10840=       IF (X(IR)) 154,157,158
10850=   154 LONG = L(IR)
10860=       IF (LRESTE-LONG.LE.0) 155,156
10870=   155 TVAL = VAL(IR) * LRESTE
10880=       TVAL = TVAL / LONG
10890=       ZBORNE = ZBORNE + TVAL
10900=       IND = IR
10910=       GOTO 160
10911=
10912=
10913=
10914=
10915=
10916=
10917=
```

```
10927=
10928=
10929=
10930=
10931=
10932=
10933=
10934=
10935=
10936=  156 ZBORNE = ZBORNE + VAL(IR)
10937=      LRESTE = LRESTE - LONG
10940=      GOTO 158
10950=  157 IF (LOGMUR) GOTO 158
10960=      ZBORNE = ZBORNE + VAL(IR)
10970=  158 CONTINUE
10980=      GOTO 399
10990=  160 ZZ = Z0 - ZBORNE
11000=      IF (ZZ.LT.10E-9) GOTO 399
11010=      IF (.NOT.LOG1) GOTO 200
11020=      IF (LOGF) GOTO 170
11030=      CALL ORDRE(QVAL,1,M)
11040=  170 DO 180 I=1,M
11050=      IND = RANG(I)
11060=      IF (X(IND).EQ.-1) GOTO 200
11070=  180 CONTINUE
11080=      WRITE (2,900)
11090=      RETURN
11100=C
11110=C   GAUCHE
11120=C
11130=  200 NIVQ = NIVQ + 1
11140=      NSOM = NSOM + 1
11150=      I1 = PT1(IND)
11160=      DEG(I1) = DEG(I1) - 1
11170=      IF ( DEG(I1).EQ.0) GOTO 210
11180=      I2 = PT2(IND)
11190=      DEG(I2) = DEG(I2) - 1
11200=      IF (DEG(I2).NE.0) GOTO 220
11210=      DEG(I2) = DEG(I2) + 1
11220=  210 DEG(I1) = DEG(I1) + 1
11230=      NSOM = NSOM + 1
11240=      PILE(NIVQ) = IND
11250=      X(IND) = 1
11260=      GOTO 20
11270=C
11280=C   ENLEVER L,ARETE IND
11290=C
11300=  220 NEL = NEL + 1
11310=      PILE(NIVQ) = -IND
11320=      X(IND) = 0
11330=      LAMBDA = LAMBDA - L(IND)
11340=      D(I1,I2) = D(I2,I1) = INFINI
11350=C
11360=C   MODIFIER C S,IL Y A LIEU
11361=
11362=
11363=
11364=
11365=
11366=
11367=
11368=
11369=
11370=
```

```
11373=
11374=
11375=
11376=
11377=
11378=
11379=
11380=
11381=C
11382=       IF (LOGMUR) GOTO 230
11390=       IF (LAMBDA.GT.0) GOTO 20
11400=       CALL FLOYD(X,IFX)
11410=       NFLOYD = NFLOYD + 1
11420=       GOTO 250
11430=C
11440=C    MODIFIER C EN APPELANT AS
11450=C
11460=   230 NM1 = N - 1
11470=       DO 232 I=1,NM1
11480=       IP1 = I + 1
11490=       DO 232 J=IP1,N
11500=   232 COP(I,J) = C(I,J)
11510=       IFXCOP = IFX
11520=       CALL AS(I1,I2,L(IND))
11530=       NAS = NAS + 1
11540=       IFX = 0
11550=       DO 235 I=1,NM1
11560=       IP1 = I+1
11570=       DO 235 J=IP1,N
11580=   235 IFX = IFX + C(I,J)
11590=       NDE(NIVO) = NDEP
11600=       IF (NDEP.GT.2000) GOTO 500
11610=       MEMC(NDEP) = IFXCOP
11620=       DO 237 I=1,NM1
11630=       IP1 = I+1
11640=       DO 237 J=IP1,N
11650=       IF ( C(I,J).EQ.COP(I,J) ) GOTO 237
11660=       NDEP = NDEP + 1
11670=       IF (NDEP.GT.2000)  GOTO 500
11680=       MEM1(NDEP) = I
11690=       MEM2(NDEP) = J
11700=       MEMC(NDEP) = COP(I,J)
11710=   237 CONTINUE
11720=       NA(NIVO) = NDEP
11730=       NDEP = NDEP + 1
11740=C    NOTE : PAS NECESSAIRE DE COPIER ET IFX CF.70
11750=C                              ...MAIS FAIT ICI POUR SIMPLIFIER LA PROGRA
MMATION
11751=
11752=
11753=
11754=
11755=
11756=
11757=
11758=
11759=
11760=
11761=
11762=
11763=
11764=
11765=
11766=
```

```
11764=
11765=
11766=
11767=
11768=
11769=
11770=
11771=       IF (LAMBDA.LE.0) GOTO 250
11781=       IF (IFX.GE.20) 400,20
11782=C  ....EN 250, ON DEVRAIT TESTER LA CONNEXITE A L'AIDE DE L'ENONCE
11790=C                              IF (IFX.GE.INFINI) GOTO 400
11800=C  ....MAIS L'ENONCE QUI SUIT LE FAIT DE TOUTE FACON
11810=   250 IF (IFX-Z0) 280,270,400
11820=   270 IF (LAMBDA.GE.LAMBDA0) GOTO 400
11830=   280 LAMBDA0 = LAMBDA    $    Z0 = IFX     $     NOPT  = NSOM
11840=       DO 281 I=1,M
11850=       X0(I) = X(I)
11860=       IF (X0(I).EQ.-1)  X0(I) = 1
11870=   281 CONTINUE
11880=       GOTO 400
11890=C
11900=C    BACKTRACKING
11910=C
11920=   399 NBORU = NBORU + 1
11930=       MCOUPE = MCOUPE + NIVO
11940=       NSOM = NSOM + 1
11950=   400 IF (NIVO.EQ.0) GOTO 490
11960=       IND = PILE(NIVO)
11970=       IF (IND.LT.0) GOTO 410
11980=       NIVO = NIVO - 1
11990=       X(IND) = -1
12000=       GOTO 400
12010=   410 IF (LOGSEV) GOTO 420
12020=       IMOD = NEL - (NEL/FREQ)*FREQ
12030=       IF(IMOD.NE.0) GOTO 430
12040=       IF(NIVO.GT.NMAX) GOTO 420
12050=       NDEB1 = ((NEL/FREQ)-1) * M
12060=       DO 415 I=1,M
12070=        ND = NDEB1+I
12080=   415 VAL(I) = VALANC(ND)
12090=       IF (LOGF) GOTO 420
12100=       DO 416 I=1,M
12110=        ND = NDEB1+I
12120=   416  OVAL(I) = OVALANC(ND)
12130=   420 NIVO = NIVO - 1
12140=       NEL  = NEL  - 1
12150=C
12151=
12152=
12153=
12154=
12155=
12156=
12157=
12158=
12159=
12160=
12161=
12162=
12163=
12164=
12165=
12166=
12167=
```

```
12187=
12188=
12189=
12190=
12191=C    DROITE
12192=C
12193=        IND = - IND
12194=        IF (LOGMUP) 460,443
12200=   443  IF (LOGSEV) GOTO 470
12210=        NIV = (NEL/FREO) + 1
12220=        IFX = MEMC(NIV)
12230=        GOTO 470
12240=   460  I1 = NDE(NIVO+1)
12250=        I2 =  NA(NIVO+1)
12260=        IFX = MEMC(I1)
12270=        I1 = I1 + 1
12280=        IF (I1.GT.I2) GOTO 470
12290=        DO 465 K=I1,I2
12300=        I = MEM1(K)
12310=        J = MEM2(K)
12320=        C(J,I) = MEMC(K)
12330=   465  C(I,J) = MEMC(K)
12340=        NDEP = I1 - 1
12350=   470  NIVO = NIVO + 1
12360=        NSOM = NSOM + 1
12370=        PILE(NIVO) = IND
12380=        X(IND) = 1
12390=        LAMBDA = LAMBDA + L(IND)
12400=        I1 = PT1(IND)  $    I2 = PT2(IND)
12410=        D(I1,I2) = D(I2,I1) =  L(IND)
12420=        DEG(I1) = DEG(I1) + 1
12430=        DEG(I2) = DEG(I2) + 1
12440=        GOTO 20
12450=   490  UTIL = NBORU * 100
12460=        UTIL = UTIL / NBORNE
12470=        RCOUPE = MCOUPE
12480=        RCOUPE = RCOUPE / NBORU
12490=        WRITE (2,491) NSOM,NAS,NFLOYD,NEVAL,NBORNE,NBORU,RCOUPE,UTIL
,NOPT
12500=   491  FORMAT (*1QUELQUES STATISTIQUES ACCUMULEES PENDANT L,EXECUTI
ON DE
12510=      -LA SOUS-ROUTINE SEARCH : *,/,/,/,* NOMBRE DE SOMMETS EXAMINE
S : *,
12520=      -I7,/,* NOMBRE DE FOIS OU LA SOUS-ROUTINE  AS  A ETE UTILISE
E : *
12530=      -I7,/,* .....FLOYD.....
. : *,
12540=      -I7,/,* NOMBRE D,EVALUATIONS DES F OU DES G : *,I7,/,* NOMB
RE D,E
12550=      -VALUATIONS DE ZBORNE :*,I7,/,* NOMBRE DE FOIS OU LE CALCUL
DE ZBO
12560=      -RNE A PERMIS DE COUPER : *,I7,/,* MCOUPE = *,F7.1,*    ...POU
RCENTA
12570=      -GE D,UTILITE DE LA BORNE : *,F7.1,/,* SOMMET OPTIMUM : *,I7
)
12580=        RETURN
12590=   500  WRITE (2,501)  NIVO ,NDEP
12600=   501  FORMAT (* DEPASSEMENT ... NIVO/NDEP : * , 2I6)
12610=   900  FORMAT (* .....ERREUR L061.....*)
12620=        RETURN
12630=        END
12640=*WEOR
01
```

LE RESEAU RH1 A 7 SOMMETS ET 16 ARETES ET IL EST DEFINI COMME SUIT :

NUMERO	PT1	PT2	LONGUEUR
1	1	2	39
2	1	3	32
3	1	4	33
4	1	5	49
5	2	4	27
6	2	5	43
7	2	6	36
8	2	7	44
9	3	4	20
10	3	5	24
11	3	6	35
12	4	5	19
13	4	6	21
14	5	6	14
15	5	7	21
16	6	7	10

exemple de sortie...

DEFINITION DES VARIABLES DE CONTROLE : LOGSG LOGSEV LOGMUR LOGF LOG1

FREQ	NMAX	IB							
			T	F	T	F	T
99	99	50					

..... LMIN = 122

.....LMAX = 467

POUR UN BUDGET DE 233
LA SOLUTION INITIALE RETIENT LES ARETES SUIVANTES :

ARETE 2
ARETE 3
ARETE 5
ARETE 9
ARETE 10
ARETE 12
ARETE 13
ARETE 14
ARETE 15
ARETE 16

Z0 = 764 LAMBDA0 = -12

TEMPS DE CALCUL DE LA SOLUTION INITIALE : 0.000 SEC.

```
QUELQUES STATISTIQUES ACCUMULEES PENDANT L'EXECUTION DE LA SOUS-ROUTINE
SEARCH :


NOMBRE DE SOMMETS EXAMINES :      175

NOMBRE DE FOIS OU LA SOUS-ROUTINE  AS  A ETE UTILISEE :       74
.....................................FLOYD..... :        1

NOMBRE D'EVALUATIONS DES F OU DES G :        1

NOMBRE D'EVALUATIONS DE ZBORNE :     101

NOMBRE DE FOIS OU LE CALCUL DE ZBORNE A PERMIS DE COUPER :       26
MCOUPE =      8.4    ...POURCENTAGE D'UTILITE DE LA BORNE :     25.7

SOMMET OPTIMUM :      113



POUR UN BUDGET DE   233
LA SOLUTION OPTIMALE RETIENT LES ARETES SUIVANTES :
                                          ARETE   3
                                          ARETE   5
                                          ARETE   7
                                          ARETE   9
                                          ARETE  10
                                          ARETE  12
                                          ARETE  13
                                          ARETE  14
                                          ARETE  15
                                          ARETE  16
                    Z0 =     761           LAMBDA0 =      -8

TEMPS DE CALCUL DE LA SOLUTION OPTIMALE :        .375    SEC.
```

APPENDICE II

Description des réseaux

RESEAU RH1
N = 7 M = 16

ARETE 1 ...DE 1 A 2 ...LONGUEUR: 39
ARETE 2 ...DE 1 A 3 ...LONGUEUR: 32
ARETE 3 ...DE 1 A 4 ...LONGUEUR: 33
ARETE 4 ...DE 1 A 5 ...LONGUEUR: 49
ARETE 5 ...DE 2 A 4 ...LONGUEUR: 27
ARETE 6 ...DE 2 A 5 ...LONGUEUR: 43
ARETE 7 ...DE 2 A 6 ...LONGUEUR: 36
ARETE 8 ...DE 2 A 7 ...LONGUEUR: 44
ARETE 9 ...DE 3 A 4 ...LONGUEUR: 20
ARETE 10 ...DE 3 A 5 ...LONGUEUR: 24
ARETE 11 ...DE 3 A 6 ...LONGUEUR: 35
ARETE 12 ...DE 4 A 5 ...LONGUEUR: 19
ARETE 13 ...DE 4 A 6 ...LONGUEUR: 21
ARETE 14 ...DE 5 A 6 ...LONGUEUR: 14
ARETE 15 ...DE 5 A 7 ...LONGUEUR: 21
ARETE 16 ...DE 6 A 7 ...LONGUEUR: 10

RESEAU RH2
N = 8 M = 18

ARETE	1	...DE	1	A 2	...LONGUEUR: 70
ARETE	2	...DE	1	A 3	...LONGUEUR: 52
ARETE	3	...DE	1	A 4	...LONGUEUR: 47
ARETE	4	...DE	2	A 3	...LONGUEUR: 36
ARETE	5	...DE	2	A 6	...LONGUEUR: 60
ARETE	6	...DE	2	A 8	...LONGUEUR: 64
ARETE	7	...DE	3	A 4	...LONGUEUR: 40
ARETE	8	...DE	3	A 5	...LONGUEUR: 72
ARETE	9	...DE	3	A 6	...LONGUEUR: 46
ARETE	10	...DE	3	A 8	...LONGUEUR: 76
ARETE	11	...DE	4	A 5	...LONGUEUR: 42
ARETE	12	...DE	4	A 6	...LONGUEUR: 68
ARETE	13	...DE	4	A 7	...LONGUEUR: 86
ARETE	14	...DE	5	A 6	...LONGUEUR: 72
ARETE	15	...DE	5	A 7	...LONGUEUR: 76
ARETE	16	...DE	6	A 7	...LONGUEUR: 24
ARETE	17	...DE	6	A 8	...LONGUEUR: 46
ARETE	18	...DE	7	A 8	...LONGUEUR: 52

RESEAU RH3
N = 9 M = 20

ARETE	1	...DE	1	A 2	...LONGUEUR: 108
ARETE	2	...DE	1	A 3	...LONGUEUR: 96
ARETE	3	...DE	1	A 4	...LONGUEUR: 99
ARETE	4	...DE	1	A 5	...LONGUEUR: 128
ARETE	5	...DE	2	A 3	...LONGUEUR: 58
ARETE	6	...DE	2	A 5	...LONGUEUR: 46
ARETE	7	...DE	2	A 9	...LONGUEUR: 86
ARETE	8	...DE	3	A 4	...LONGUEUR: 12
ARETE	9	...DE	3	A 5	...LONGUEUR: 40
ARETE	10	...DE	3	A 7	...LONGUEUR: 58
ARETE	11	...DE	3	A 8	...LONGUEUR: 42
ARETE	12	...DE	4	A 6	...LONGUEUR: 62
ARETE	13	...DE	4	A 7	...LONGUEUR: 62
ARETE	14	...DE	4	A 8	...LONGUEUR: 38
ARETE	15	...DE	5	A 6	...LONGUEUR: 21
ARETE	16	...DE	5	A 8	...LONGUEUR: 48
ARETE	17	...DE	6	A 7	...LONGUEUR: 15
ARETE	18	...DE	6	A 9	...LONGUEUR: 32
ARETE	19	...DE	7	A 8	...LONGUEUR: 38
ARETE	20	...DE	7	A 9	...LONGUEUR: 34

RESEAU RB4
N = 10 M = 45

ARETE 1 ...DE 1 A 2 ...LONGUEUR: 470
ARETE 2 ...DE 1 A 3 ...LONGUEUR: 519
ARETE 3 ...DE 1 A 4 ...LONGUEUR: 236
ARETE 4 ...DE 1 A 5 ...LONGUEUR: 369
ARETE 5 ...DE 1 A 6 ...LONGUEUR: 94
ARETE 6 ...DE 1 A 7 ...LONGUEUR: 383
ARETE 7 ...DE 1 A 8 ...LONGUEUR: 371
ARETE 8 ...DE 1 A 9 ...LONGUEUR: 300
ARETE 9 ...DE 1 A 10 ...LONGUEUR: 547
ARETE 10 ...DE 2 A 3 ...LONGUEUR: 982
ARETE 11 ...DE 2 A 4 ...LONGUEUR: 281
ARETE 12 ...DE 2 A 5 ...LONGUEUR: 143
ARETE 13 ...DE 2 A 6 ...LONGUEUR: 488
ARETE 14 ...DE 2 A 7 ...LONGUEUR: 851
ARETE 15 ...DE 2 A 8 ...LONGUEUR: 98
ARETE 16 ...DE 2 A 9 ...LONGUEUR: 296
ARETE 17 ...DE 2 A 10 ...LONGUEUR: 393
ARETE 18 ...DE 3 A 4 ...LONGUEUR: 750
ARETE 19 ...DE 3 A 5 ...LONGUEUR: 864
ARETE 20 ...DE 3 A 6 ...LONGUEUR: 539
ARETE 21 ...DE 3 A 7 ...LONGUEUR: 204
ARETE 22 ...DE 3 A 8 ...LONGUEUR: 885
ARETE 23 ...DE 3 A 9 ...LONGUEUR: 801
ARETE 24 ...DE 3 A 10 ...LONGUEUR: 934
ARETE 25 ...DE 4 A 5 ...LONGUEUR: 246
ARETE 26 ...DE 4 A 6 ...LONGUEUR: 220
ARETE 27 ...DE 4 A 7 ...LONGUEUR: 594
ARETE 28 ...DE 4 A 8 ...LONGUEUR: 191
ARETE 29 ...DE 4 A 9 ...LONGUEUR: 81
ARETE 30 ...DE 4 A 10 ...LONGUEUR: 523
ARETE 31 ...DE 5 A 6 ...LONGUEUR: 410
ARETE 32 ...DE 5 A 7 ...LONGUEUR: 750
ARETE 33 ...DE 5 A 8 ...LONGUEUR: 95
ARETE 34 ...DE 5 A 9 ...LONGUEUR: 298
ARETE 35 ...DE 5 A 10 ...LONGUEUR: 297
ARETE 36 ...DE 6 A 7 ...LONGUEUR: 374
ARETE 37 ...DE 6 A 8 ...LONGUEUR: 391
ARETE 38 ...DE 6 A 9 ...LONGUEUR: 262
ARETE 39 ...DE 6 A 10 ...LONGUEUR: 622
ARETE 40 ...DE 7 A 8 ...LONGUEUR: 752
ARETE 41 ...DE 7 A 9 ...LONGUEUR: 633
ARETE 42 ...DE 7 A 10 ...LONGUEUR: 881
ARETE 43 ...DE 8 A 9 ...LONGUEUR: 224
ARETE 44 ...DE 8 A 10 ...LONGUEUR: 389
ARETE 45 ...DE 9 A 10 ...LONGUEUR: 588

RESEAU R5
N = 12 M = 20

ARETE	1	...DE	1	A	5	...LONGUEUR: 25
ARETE	2	...DE	1	A	7	...LONGUEUR: 126
ARETE	3	...DE	1	A	8	...LONGUEUR: 170
ARETE	4	...DE	1	A	10	...LONGUEUR: 24
ARETE	5	...DE	1	A	11	...LONGUEUR: 117
ARETE	6	...DE	2	A	5	...LONGUEUR: 117
ARETE	7	...DE	2	A	6	...LONGUEUR: 200
ARETE	8	...DE	2	A	10	...LONGUEUR: 109
ARETE	9	...DE	2	A	11	...LONGUEUR: 132
ARETE	10	...DE	3	A	9	...LONGUEUR: 32
ARETE	11	...DE	3	A	11	...LONGUEUR: 60
ARETE	12	...DE	3	A	12	...LONGUEUR: 40
ARETE	13	...DE	4	A	8	...LONGUEUR: 250
ARETE	14	...DE	4	A	9	...LONGUEUR: 48
ARETE	15	...DE	4	A	11	...LONGUEUR: 79
ARETE	16	...DE	6	A	11	...LONGUEUR: 80
ARETE	17	...DE	6	A	12	...LONGUEUR: 50
ARETE	18	...DE	7	A	8	...LONGUEUR: 65
ARETE	19	...DE	10	A	11	...LONGUEUR: 88
ARETE	20	...DE	11	A	12	...LONGUEUR: 132

RESEAU R6
N = 20 M = 40

ARETE 1 ...DE 1 A 2 ...LONGUEUR: 4
ARETE 2 ...DE 1 A 3 ...LONGUEUR: 3
ARETE 3 ...DE 1 A 4 ...LONGUEUR: 4
ARETE 4 ...DE 1 A 5 ...LONGUEUR: 5
ARETE 5 ...DE 2 A 3 ...LONGUEUR: 5
ARETE 6 ...DE 2 A 5 ...LONGUEUR: 6
ARETE 7 ...DE 2 A 6 ...LONGUEUR: 4
ARETE 8 ...DE 3 A 4 ...LONGUEUR: 6
ARETE 9 ...DE 3 A 14 ...LONGUEUR: 20
ARETE 10 ...DE 3 A 15 ...LONGUEUR: 9
ARETE 11 ...DE 4 A 5 ...LONGUEUR: 8
ARETE 12 ...DE 4 A 12 ...LONGUEUR: 12
ARETE 13 ...DE 4 A 13 ...LONGUEUR: 18
ARETE 14 ...DE 5 A 6 ...LONGUEUR: 10
ARETE 15 ...DE 5 A 7 ...LONGUEUR: 7
ARETE 16 ...DE 5 A 11 ...LONGUEUR: 15
ARETE 17 ...DE 5 A 12 ...LONGUEUR: 13
ARETE 18 ...DE 6 A 7 ...LONGUEUR: 6
ARETE 19 ...DE 6 A 8 ...LONGUEUR: 8
ARETE 20 ...DE 6 A 17 ...LONGUEUR: 8
ARETE 21 ...DE 7 A 8 ...LONGUEUR: 3
ARETE 22 ...DE 7 A 9 ...LONGUEUR: 15
ARETE 23 ...DE 8 A 9 ...LONGUEUR: 22
ARETE 24 ...DE 8 A 18 ...LONGUEUR: 15
ARETE 25 ...DE 9 A 10 ...LONGUEUR: 9
ARETE 26 ...DE 9 A 20 ...LONGUEUR: 30
ARETE 27 ...DE 10 A 11 ...LONGUEUR: 45
ARETE 28 ...DE 11 A 12 ...LONGUEUR: 10
ARETE 29 ...DE 12 A 13 ...LONGUEUR: 15
ARETE 30 ...DE 13 A 14 ...LONGUEUR: 28
ARETE 31 ...DE 13 A 15 ...LONGUEUR: 35
ARETE 32 ...DE 14 A 15 ...LONGUEUR: 20
ARETE 33 ...DE 15 A 16 ...LONGUEUR: 20
ARETE 34 ...DE 16 A 17 ...LONGUEUR: 10
ARETE 35 ...DE 16 A 19 ...LONGUEUR: 20
ARETE 36 ...DE 17 A 18 ...LONGUEUR: 16
ARETE 37 ...DE 17 A 19 ...LONGUEUR: 18
ARETE 38 ...DE 18 A 19 ...LONGUEUR: 25
ARETE 39 ...DE 18 A 20 ...LONGUEUR: 27
ARETE 40 ...DE 19 A 20 ...LONGUEUR: 35

RESEAU R7
N = 20 M = 40

ARETE	1	...DE	1	A	7	...LONGUEUR: 25
ARETE	2	...DE	1	A	12	...LONGUEUR: 126
ARETE	3	...DE	1	A	13	...LONGUEUR: 170
ARETE	4	...DE	1	A	17	...LONGUEUR: 24
ARETE	5	...DE	1	A	18	...LONGUEUR: 117
ARETE	6	...DE	2	A	4	...LONGUEUR: 30
ARETE	7	...DE	2	A	6	...LONGUEUR: 34
ARETE	8	...DE	2	A	10	...LONGUEUR: 35
ARETE	9	...DE	2	A	15	...LONGUEUR: 37
ARETE	10	...DE	2	A	20	...LONGUEUR: 36
ARETE	11	...DE	3	A	7	...LONGUEUR: 117
ARETE	12	...DE	3	A	8	...LONGUEUR: 200
ARETE	13	...DE	3	A	17	...LONGUEUR: 109
ARETE	14	...DE	3	A	18	...LONGUEUR: 132
ARETE	15	...DE	4	A	14	...LONGUEUR: 32
ARETE	16	...DE	4	A	18	...LONGUEUR: 60
ARETE	17	...DE	4	A	19	...LONGUEUR: 24
ARETE	18	...DE	4	A	20	...LONGUEUR: 40
ARETE	19	...DE	5	A	13	...LONGUEUR: 250
ARETE	20	...DE	5	A	14	...LONGUEUR: 48
ARETE	21	...DE	5	A	16	...LONGUEUR: 64
ARETE	22	...DE	5	A	18	...LONGUEUR: 79
ARETE	23	...DE	6	A	10	...LONGUEUR: 40
ARETE	24	...DE	6	A	11	...LONGUEUR: 33
ARETE	25	...DE	6	A	19	...LONGUEUR: 43
ARETE	26	...DE	8	A	9	...LONGUEUR: 59
ARETE	27	...DE	8	A	15	...LONGUEUR: 85
ARETE	28	...DE	8	A	18	...LONGUEUR: 80
ARETE	29	...DE	8	A	20	...LONGUEUR: 50
ARETE	30	...DE	9	A	15	...LONGUEUR: 41
ARETE	31	...DE	9	A	20	...LONGUEUR: 21
ARETE	32	...DE	10	A	15	...LONGUEUR: 26
ARETE	33	...DE	11	A	16	...LONGUEUR: 16
ARETE	34	...DE	11	A	19	...LONGUEUR: 45
ARETE	35	...DE	12	A	13	...LONGUEUR: 65
ARETE	36	...DE	14	A	19	...LONGUEUR: 35
ARETE	37	...DE	15	A	20	...LONGUEUR: 50
ARETE	38	...DE	16	A	19	...LONGUEUR: 24
ARETE	39	...DE	17	A	18	...LONGUEUR: 88
ARETE	40	...DE	18	A	20	...LONGUEUR: 132

RESEAU R8
N = 29 M = 54

ARETE 1 ...DE 1 A 8 ...LONGUEUR: 96
ARETE 2 ...DE 1 A 9 ...LONGUEUR: 39
ARETE 3 ...DE 1 A 14 ...LONGUEUR: 115
ARETE 4 ...DE 1 A 15 ...LONGUEUR: 32
ARETE 5 ...DE 2 A 18 ...LONGUEUR: 63
ARETE 6 ...DE 2 A 23 ...LONGUEUR: 48
ARETE 7 ...DE 2 A 26 ...LONGUEUR: 95
ARETE 8 ...DE 3 A 7 ...LONGUEUR: 117
ARETE 9 ...DE 3 A 17 ...LONGUEUR: 109
ARETE 10 ...DE 3 A 18 ...LONGUEUR: 132
ARETE 11 ...DE 3 A 22 ...LONGUEUR: 69
ARETE 12 ...DE 4 A 18 ...LONGUEUR: 94
ARETE 13 ...DE 4 A 19 ...LONGUEUR: 24
ARETE 14 ...DE 4 A 25 ...LONGUEUR: 32
ARETE 15 ...DE 4 A 27 ...LONGUEUR: 40
ARETE 16 ...DE 4 A 29 ...LONGUEUR: 30
ARETE 17 ...DE 5 A 12 ...LONGUEUR: 207
ARETE 18 ...DE 5 A 16 ...LONGUEUR: 45
ARETE 19 ...DE 6 A 10 ...LONGUEUR: 40
ARETE 20 ...DE 6 A 11 ...LONGUEUR: 33
ARETE 21 ...DE 6 A 19 ...LONGUEUR: 43
ARETE 22 ...DE 6 A 29 ...LONGUEUR: 34
ARETE 23 ...DE 7 A 21 ...LONGUEUR: 25
ARETE 24 ...DE 8 A 14 ...LONGUEUR: 124
ARETE 25 ...DE 8 A 15 ...LONGUEUR: 125
ARETE 26 ...DE 9 A 15 ...LONGUEUR: 41
ARETE 27 ...DE 9 A 24 ...LONGUEUR: 59
ARETE 28 ...DE 9 A 27 ...LONGUEUR: 21
ARETE 29 ...DE 10 A 15 ...LONGUEUR: 26
ARETE 30 ...DE 10 A 29 ...LONGUEUR: 35
ARETE 31 ...DE 11 A 19 ...LONGUEUR: 45
ARETE 32 ...DE 11 A 26 ...LONGUEUR: 16
ARETE 33 ...DE 12 A 13 ...LONGUEUR: 65
ARETE 34 ...DE 12 A 28 ...LONGUEUR: 39
ARETE 35 ...DE 13 A 16 ...LONGUEUR: 173
ARETE 36 ...DE 13 A 21 ...LONGUEUR: 170
ARETE 37 ...DE 15 A 24 ...LONGUEUR: 85
ARETE 38 ...DE 15 A 27 ...LONGUEUR: 50
ARETE 39 ...DE 15 A 29 ...LONGUEUR: 37
ARETE 40 ...DE 17 A 18 ...LONGUEUR: 88
ARETE 41 ...DE 17 A 21 ...LONGUEUR: 24
ARETE 42 ...DE 18 A 21 ...LONGUEUR: 117
ARETE 43 ...DE 18 A 22 ...LONGUEUR: 117
ARETE 44 ...DE 18 A 23 ...LONGUEUR: 79
ARETE 45 ...DE 18 A 24 ...LONGUEUR: 80
ARETE 46 ...DE 18 A 27 ...LONGUEUR: 132
ARETE 47 ...DE 19 A 25 ...LONGUEUR: 35
ARETE 48 ...DE 19 A 26 ...LONGUEUR: 24
ARETE 49 ...DE 20 A 21 ...LONGUEUR: 67
ARETE 50 ...DE 20 A 28 ...LONGUEUR: 20
ARETE 51 ...DE 23 A 25 ...LONGUEUR: 48
ARETE 52 ...DE 23 A 26 ...LONGUEUR: 64
ARETE 53 ...DE 24 A 27 ...LONGUEUR: 50
ARETE 54 ...DE 27 A 29 ...LONGUEUR: 36

REMERCIEMENTS

Pendant les deux dernières années, j'étais membre du Centre de recherche sur les transports à titre de chercheur. J'y ai trouvé une ambiance agréable et propice au travail de recherche.

Je remercie vivement Monsieur Michael Florian, directeur du Centre, qui, à titre de directeur de thèse, m'a guidé et stimulé tout au long de mon travail. J'ai particulièrement apprécié la critique constructive qu'il a faite de mes divers manuscrits. Je remercie aussi Monsieur Pierre Robillard de l'intérêt qu'il a porté à mon travail et de l'aide bienveillante qu'il m'a apporté en tant que membre du conseil de direction du Centre.

Mes remerciements vont aussi à mes collègues du Centre de recherche sur les transports,

Claude A., qui m'a fait voir ses photos de vacances pour me consoler de ne pas en avoir (de vacances),

Claude C., qui m'a servi un délicieux couscous pour bien commencer la dernière semaine où j'ai travaillé sur ma thèse,

Michel T., qui m'a dépanné dans un moment difficile (mathématiquement parlant),

Robert C., dont les connaissances en combinatorique ont contribué à me faire mieux comprendre un certain aspect de mon travail,

Sang N., qui m'a aidé à situer mon sujet dans le contexte général des problèmes de transport.

Madame Jacqueline Gariépy s'est chargée de la dactylographie. Elle a fait de l'excellent travail. Je lui exprime ma reconnaissance.

Je mentionne enfin l'aide financière que j'ai reçue du Conseil des arts du Canada, à titre de boursier, depuis le début de mes études de doctorat (septembre 1970) jusqu'en décembre de l'an dernier.

BIBLIOGRAPHIE

Barbier, M. (1966)
LE FUTUR RESEAU DE TRANSPORT EN REGION DE PARIS
Cahiers de l'Institut d'Aménagements et d'Urbanisme de la Région Parisienne, vol. IV - V, no 4.

Berge, C. (1967)
THEORIE DES GRAPHES ET SES APPLICATIONS
Dunod, Paris, 1967.

Billheimer, J.W. et Gray, P. (1973)
NETWORK DESIGN WITH FIXED AND VARIABLE COST ELEMENTS
Transportation Science, vol. 7 no 1, pp. 49-74.

Boyce, D.E., Farhi, A. et Weischedel, R. (1973)
OPTIMAL NETWORK PROBLEM: A BRANCH-AND-BOUND ALGORITHM
Environment and Planning, vol. 5, pp. 519-533.

Bruynooghe, M. (1972)
AN OPTIMAL METHOD OF CHOICE OF INVESTMENTS IN A TRANSPORT NETWORK
Paper presented at the Planning and Transport Research and Computation Seminars on Urban Traffic Model Research, London (8-12 May 1972).

Carter, E.C. et Stowers, J.R. (1973)
MODEL FOR FUNDS ALLOCATION FOR URBAN HIGHWAY SYSTEM CAPACITY IMPROVEMENTS
Highway Research Record no 20, Washington D.C., pp. 84-102.

Chan, Y.-P. (1969)
OPTIMAL TRAVEL TIME REDUCTION IN A TRANSPORT NETWORK: AN APPLICATION OF NETWORK AGGREGATION AND BRANCH-AND-BOUND TECHNIQUES
Research Report R69-39, M.I.T., Department of Divil Engineering, Cambridge, Mass.

Dafermos, S.C. (1973)
TOLL PATTERNS FOR MULTICLASS-USER TRANSPORTATION NETWORKS
Transportation Science, vol. 7 no 3, pp. 211-223.

Dantzig, G.B. (1957)
DISCRETE VARIABLE EXTREMUM PROBLEMS
Operations Research, vol. 5 no 2, pp. 266-277.

Dijkstra, E.W. (1959)
A NOTE ON TWO PROBLEMS IN CONNECTION WITH GRAPHS
Numerische Mathematik, 1, pp. 269-271.

Dreyfus, S.E. (1969)
AN APPRAISAL OF SOME SHORTEST-PATH ALGORITHMS
Operations Research, vol. 7 no 3, pp. 395-412.

Floyd, R.W. (1962)
ALGORITHM 97 : SHORTEST PATH
Comm. of ACM 5 (1962), p. 345.

Hershdorfer, A.M. (1965)
OPTIMAL UTILISATION AND SYNTHESIS OF ROAD NETWORKS
M.I.T., Department of Civil Engineering, Cambridge, Mass.

Hoang, H.H. (1971)
A COMPUTATIONAL APPROACH TO THE SELECTION OF AN OPTIMAL NETWORK
Document de travail # 10, département d'informatique,
Université de Montréal, août 1971.
(Note: publié aussi dans Management Science, vol. 19 no 5,
janvier 1973).

Jorgensen, N.O. (1963)
SOME ASPECTS OF THE URBAN TRAFFIC ASSIGNMENT PROBLEM
Graduate Report, Institute of Transportation and Traffic Engineering,
University of California, Berkeley.

Lawler, E.L., and Wood, D.E. (1966)
BRANCH-AND-BOUND METHODS: A SURVEY
Operations Research, vol. 14 no 4, pp. 699-719.

Leblanc, L.J. (1973)
MATHEMATICAL PROGRAMMING ALGORITHMS FOR LARGE SCALE NETWORK EQUILIBRIUM AND NETWORK DESIGN PROBLEMS
Thèse de doctorat, Department of Industrial Engineering and Management Sciences, Northwestern University, Illinois.

Loubal, P.S. (1967)
A NETWORK EVALUATION PROCEDURE
Highway Research Record no 205, Washington D.C.

Murchland, J.D. (1970)
A FIXED MATRIX METHOD FOR ALL SHORTEST DISTANCES IN A DIRECTED GRAPH AND FOR THE INVERSE PROBLEM
Thèse de doctorat, Université de Karlsruhe, 59 p.

Ochoa-Rosso, F. (1968)
APPLICATIONS OF DISCRETE OPTIMIZATION TECHNIQUES TO CAPITAL INVESTMENT AND NETWORK SYNTHESIS PROBLEMS
Research Report R68-40, M.I.T., Department of Civil Engineering, Cambridge, Mass.

Ochoa-Rosso, F. et Silva, A. (1968)
OPTIMUM PROJECT ADDITION IN URBAN TRANSPORTATION NETWORKS VIA DESCRIPTIVE ASSIGNMENT MODELS
Research Report R68-44, M.I.T., Department of Civil Engineering, Cambridge, Mass.

Ridley, T.M. (1968)
AN INVESTMENT POLICY TO REDUCE THE TRAVEL TIME IN A TRANSPORTATION NETWORK
Transportation Research, vol. 2 no 4, pp. 409-424.

Roberts, P.O. et Funk, M.L. (1964)
TOWARD OPTIMUM METHODS OF LINK ADDITION IN TRANSPORTATION NETWORKS
M.I.T., Department of Civil Engineering, Cambridge, Mass.

Roy, B. (1969)
PROCEDURE D'EXPLORATION PAR SEPARATION ET EVALUATION
R.I.R.O., 3e année, no V - 1, pp. 61-90.

Schwartz, J.G. (1968)
NETWORK FLOW AND SYNTHESIS MODELS FOR TRANSPORTATION PLANNING:
A SURVEY
Research Report R68-49, M.I.T., Department of Civil Engineering,
Cambridge, Mass.

Scott, A.J. (1967)
A PROGRAMMING MODEL OF AN INTEGRATED TRANSPORTATION NETWORK
The Regional Science Association Papers, vol. 19.

Scott, A.J. (1969)
THE OPTIMAL NETWORK PROBLEM: SOME COMPUTATIONAL PROCEDURES
Transportation Research, vol. 3 no 2, pp. 201-210.

Simonnard, M.
PROGRAMMATION LINEAIRE
Dunod, Paris, 1962.

Stairs, S. (1968)
SELECTING AN OPTIMAL TRAFFIC NETWORK
Journal of Transport Economics and Policy, vol. 2, pp. 218-231.

Steenbrink, P.A. (1974)
OPTIMIZATION OF TRANSPORT NETWORKS
John Wiley & Sons, London (New York, Sydney, Toronto).

Tomlin, J.A. (1966)
MINIMUM-COST MULTICOMMODITY NETWORK FLOWS
Operations Research, vol. 14 no 1, pp. 45-51.

Wardrop, J.G. (1952)
SOME THEORETICAL ASPECTS OF ROAD TRAFFIC RESEARCH
Proceedings, Institute of Civil Engineers, Part II, vol. 1,
pp. 325-362.